

Informační systém pro řízení schvalovacího procesu objednávek služeb a zboží

Bc. Petr Lúsar

Diplomová práce
2016



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2015/2016

ZADÁNÍ DIPLOMOVÉ PRÁCE (PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Petr Luser**
Osobní číslo: **A14548**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **kombinovaná**

Téma práce: **Informační systém pro řízení schvalovacího procesu objednávek služeb a zboží**

Téma anglicky: **An Information System for Goods and Services Order Approval Management**

Zásady pro vypracování:

1. Popište současný stav schvalovacího procesu objednávek služeb a zboží ve Vsetínské nemocnici a.s. a shrňte požadavky na nový informační systém pro podporu tohoto procesu.
2. Vyberte vhodné technologie k implementaci systému a popište je.
3. Vypracujte diagram případů užití a navrhnete relační model databáze.
4. Vytvořte uživatelskou aplikaci informačního systému a otestujte ji na zkušebních datech.
5. Navrhnete a provedte potřebná zabezpečení.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. KRAVAL, Ilja. Analytické modelování informačních systémů pomocí UML v praxi. 2010. Lipina: Object Consulting, 2010. ISBN 978-80-254-6986-6.
2. PHP5, MySQL, Apache: vytváříme webové aplikace. Vyd. 1. Brno: Computer Press, 2006, 813 s. ISBN 80-251-1073-7.
3. PHP Documentation [online]. PHP Group, 2015, poslední aktualizace 3.12.2015. Dostupné z: <https://secure.php.net/docs.php>
4. MySQL Documentation [online]. Oracle Corporation, 2015. Dostupné z: <http://dev.mysql.com/doc/>
5. Dokumentace Nette Framework. Nette Framework [online]. Nette Foundation, 2015, poslední aktualizace 18.8.2015. Dostupné z: <https://doc.nette.org/cs/2.3/>.

Vedoucí diplomové práce:

Ing. Radek Vala, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

5. února 2016

Termín odevzdání diplomové práce:

20. května 2016

Ve Zlíně dne 5. února 2016



doc. Mgr. Milan Adámek, Ph.D.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu


Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 5. února 2016


.....
podpis diplomanta

ABSTRAKT

Tato práce se zabývá analýzou, návrhem a implementací informačního systému určeného pro řízení schvalovacího procesu objednávek služeb a zboží. Vytvořený informační systém poskytuje úplnou podporu pro řešený podnikový proces v souladu s vnitropodnikovou směrnicí. Systém je implementován jako webová aplikace v jazyce PHP s využitím Nette Frameworku. Jako databázový server je využit MySQL.

Klíčová slova: PHP, Nette Framework, MySQL, podnikový proces, UML diagram

ABSTRACT

This thesis deals with the analysis, design and implementation of the information system which should be used for management of approval process for goods and services orders. This information system provides full support for this concrete company process and it is in obedience to an internal regulation. The system is implemented as a web application in PHP using Nette Framework. As the database server is used MySQL.

Keywords: PHP, Nette Framework, MySQL, Business Process, UML diagram

Upřímně rád bych zde chtěl poděkovat vedoucímu své práce, panu Ing. Radku Valovi, Ph.D., za odbornou pomoc, cenné rady a připomínky, které mi pomohly při tvorbě diplomové práce. Vážím si též jeho laskavého přístupu, vstřícnosti a profesionality, jimiž vedení této práce obohatil.

Současně bych velmi rád poděkoval i své ženě a dětem za trpělivost a podporu, kterou mi po dobu mého studia nezištně poskytovali.

OBSAH

ÚVOD	10
I TEORETICKÁ ČÁST	11
1 ANALÝZA SOUČASNÉHO STAVU	12
1.1 OBJEDNÁVKA	12
1.1.1 Postup zpracování požadavku	13
2 ANALÝZA POŽADAVKŮ NA INFORMAČNÍ SYSTÉM	14
2.1 DOSTUPNÉ MOŽNOSTI ŘEŠENÍ.....	15
2.1.1 Existující řešení	15
CRM systémy.....	15
Microsoft SharePoint	15
2.1.2 Framework	16
2.1.3 Zhodnocení.....	16
3 TECHNOLOGIE PRO TVORBU APLIKAČNÍ ČÁSTI	18
3.1 PHP.....	18
3.1.1 Historie	18
3.1.2 Výhody.....	19
3.2 MYSQL	19
3.2.1 Formát MyISAM.....	19
3.2.2 Formát InnoDB	20
3.3 MYSQL WORKBENCH	20
3.4 ENTERPRISE ARCHITECT	21
3.4.1 UML.....	22
3.4.2 Business Process Diagram	22
3.4.3 Model případů užití	23
3.5 ARCHITEKTURA MVC.....	23
3.5.1 Model	24
3.5.2 View	24
3.5.3 Controller	24
3.6 ARCHITEKTURA MVP	24
3.7 NETTE FRAMEWORK	25
3.7.1 Dependency Injection.....	26
3.7.2 Vrstva Presenter	27
3.7.3 Zpracování akce presenteru.....	27
3.7.4 Životní cyklus presenteru	28
3.7.5 Přístup k databázi	29
3.7.6 Šablonovací systém Latte.....	31
3.7.7 Ochrana escapováním	32
3.7.8 Konfigurace.....	33
3.7.9 Formuláře	33
3.8 TECHNOLOGIE PRO TVORBU FRONTEND ČÁSTI	34
3.8.1 Twitter Bootstrap	34
3.8.2 Knihovny jQuery a jQueryUI.....	35
II PRAKTICKÁ ČÁST	36

4	ANALÝZA A NÁVRH ŘEŠENÍ.....	37
4.1	MODELOVÁNÍ PODNIKOVÉHO PROCESU	37
4.1.1	Hlavní tok.....	37
4.1.2	Diagram atomických aktivit.....	39
4.1.3	Proces realizace objednávky	42
4.2	FUNKČNÍ POŽADAVKY	43
4.3	NEFUNKČNÍ POŽADAVKY	43
4.4	DIAGRAMY PŘÍPADŮ UŽITÍ A SCÉNÁŘE	43
4.4.1	UC01: Založení nového uživatele.....	48
4.4.2	UC02: Přiřazení role	49
4.4.3	UC03: Přiřazení podřízeného oddělení	49
4.4.4	UC04: Vyžádání zapomenutého hesla	49
4.4.5	UC05: Změna hesla.....	50
4.4.6	UC06: Přiřazení odborné oblasti.....	50
4.4.7	UC07: Editace uživatele.....	50
4.4.8	UC08: Odebrání role	50
4.4.9	UC12: Odebrání odborné oblasti	51
4.4.10	UC13: Vytvoření objednávky	51
4.4.11	UC14: Zobrazení objednávky	52
4.4.12	UC15: Vložení dokumentu	52
4.4.13	UC16: Vložení komentáře.....	52
4.4.14	UC17: Vložení stanoviska vedoucího oddělení	52
4.4.15	UC18: Zaslání informace o změně stavu	53
4.4.16	UC19: Vložení odborného posudku.....	53
4.4.17	UC20: Zařazení do odborné oblasti	54
4.4.18	UC21: Vložení stanoviska.....	54
4.4.19	UC22: Vytvoření konzultace.....	54
4.4.20	UC23: Vložení příspěvku.....	55
4.4.21	UC24: Zaslání informace o požadavku na konzultaci	55
4.4.22	UC25: Správa číselníků.....	56
4.4.23	UC26: Vložení záznamu	56
4.4.24	UC27: Editace záznamu	56
5	NÁVRH ZÁKLADNÍCH ENTIT	57
5.1	OBJEDNÁVKA A JEJÍ POLOŽKY	57
5.2	ODBORNÁ OBLAST.....	57
5.3	ODBORNÉ POSUDKY OBJEDNÁVKY	58
5.4	STANOVISKA K OBJEDNÁVCE.....	58
5.5	DOKUMENTY	59
5.6	ZÁZNAM O REALIZACI OBJEDNÁVKY	60
5.7	KONZULTACE K OBJEDNÁVCE	60
5.8	UŽIVATEL A UŽIVATELSKÉ ROLE	61
5.9	ORGANIZAČNÍ ČLENĚNÍ	61
6	ROZŠÍŘENÝ ENTITNĚ RELAČNÍ MODEL DATABÁZE	62
7	NÁVRH UŽIVATELSKÉHO ROZHŘANÍ APLIKACE	68

7.1	PŘIHLÁŠENÍ	69
7.2	Hlavní strana	70
7.3	Hlavní menu	70
7.4	Drobečková navigace	71
7.5	DataGrid Grido	71
7.6	Záznam objednávky	72
7.7	Detail položky objednávky	74
8	NÁVRH STRUKTURY APLIKACE	76
8.1	Modelová vrstva	76
8.2	Řídící vrstva	77
9	ZABEZPEČENÍ	78
9.1	Řízení uživatelských oprávnění	78
9.2	Bezpečné uložení hesla	81
	ZÁVĚR	82
	SEZNAM POUŽITÉ LITERATURY	84
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	88
	SEZNAM OBRÁZKŮ	90
	SEZNAM PŘÍLOH	92

ÚVOD

Téměř v každé oblasti lidské činnosti se setkáváme s potřebou zpracování informací. Jde-li o každodenní drobná rozhodnutí, vystačíme si bezesporu bez technické podpory. Pokud však řešíme složitější úlohy v oblasti podnikových procesů, pak se již mnohdy neobejdeme bez prostředků, které nám umožní informace uchovávat, zpracovávat a třídit za účelem rozhodování a řízení. Vzniká tak zřejmá potřeba provozovat informační systém pro podporu těchto procesů.

Ve Vsetínské nemocnici a.s. existuje deficit technické podpory v oblasti řízení schvalovacího procesu objednávek služeb a zboží. Ze strany vedení nemocnice tak vzešel požadavek na vytvoření informačního systému, který by pokrýval příslušnou problematiku za podmínky zachování business procesů, vymezených vnitropodnikovou směrnicí.

Cílem této práce bude navrhnout a implementovat informační systém, který zajistí tok požadavků od základních oddělení směrem k rozhodovacím článkům firmy, na základě kterých bude možné přijímat rozhodnutí o nákupu. Úlohou informačního systému bude poskytovat podporu schvalovacímu procesu a zpracovávané informace uchovávat a předkládat jednotlivým účastníkům zpracování úplně a průkazně.

Při návrhu aplikace budou použity moderní technologie, jako jsou skriptovací jazyk PHP, databázový server MySQL, frameworky Nette, jQuery a Twitter Bootstrap. Informační systém bude navržen jako modulární v architektuře MVC.

I. TEORETICKÁ ČÁST

1 ANALÝZA SOUČASNÉHO STAVU

Schvalovací proces objednávek služeb a zboží je upraven podnikovou směrnicí Nakupování č. SM – 36-0200/09. Směrnice stanovuje závazný postup nemocnice při zadávání veřejných zakázek vypisovaných na základě zákona č. 137/2006 Sb., o zadávání veřejných zakázek, ve znění pozdějších předpisů a dále stanovuje jednotný postup pro nakupování prostředků, zboží a služeb pro potřeby Vsetínské nemocnice a.s., které nepodléhají zákonu o zadávání veřejných zakázek. Směrnice je závazná pro všechny zaměstnance nemocnice, zejména pro ty zaměstnance, kterým jsou přiděleny kompetence při nakupování. Konkrétní odpovědnost, včetně vymezení spolupráce, je uvedena při jednotlivých činnostech v rámci zajišťování procesu nakupování.

1.1 Objednávka

Požadavek na nákup zboží či služeb se dle odst. 2 čl. 8 směrnice předkládá na formuláři Objednávka zboží/služeb, který je uveden v příloze (Příloha P I). Formulář se používá ve všech případech, kdy se jedná o nákup zboží charakteru DHM¹ nebo DDHM² a u ostatních prostředků a služeb (hmotných i nehmotných). Zaměstnanci požadující nákup prostředků zajistí zpracování svého požadavku do konkrétního návrhu Objednávky zboží/služeb. V objednávce se povinně uvedou identifikační údaje požadujícího a přesná specifikace předmětu nákupu, včetně požadavků na kvalitativní ukazatele. Požadavky na nákup jsou uplatňovány, po předchozím písemném schválení oprávněným zaměstnancem, jednotlivými žadateli u zaměstnance příslušného oddělení odpovědného za nákup.

Směrnice rozděluje nakupované zboží, prostředky a služby do jednotlivých skupin. Pro každou skupinu jsou v čl. 9 směrnice definováni zaměstnanci, kteří zajišťují činnosti spojené s nákupem. Činnostmi, ve smyslu směrnice, se rozumí:

- Zodpovídá/Přezkoumává,
- Schvaluje,
- Nakupuje.

¹ DHM – dlouhodobý hmotný majetek

² DDHM – dlouhodobý drobný hmotný majetek

Ukázka definice pověřených zaměstnanců pro jednotlivé okruhy předmětu nákupu a činnosti je uvedena na obrázku (Obr. 1).

Položka	Oddělení	Zodpovídá/přezkoumává	Schvaluje	Nakupuje
3k	hardware a software (DDHM a DDNM), materiál pro modernizaci, údržbu a opravy výpočetní techniky			
	Ekonomicko obchodní odbor (EEO)	vedoucí OIS	náměstek pro vnitřní služby	pověřený zaměstnanec centrálního skladu

Obr. 1. Vzor definice pověřených zaměstnanců pro jednotlivé činnosti zpracování objednávky

1.1.1 Postup zpracování požadavku

Přesná specifikace požadavku na nákup patří k nejdůležitějším úkonům při zpracování každého požadavku v písemné objednávce. Je jednou z hlavních podmínek pro zajištění kvalitního vstupu. Tuto specifikaci uvede písemně jmenovitým výčtem pověřený zaměstnanec oddělení, pro které je nákup realizován, a to zaměstnanci odpovědnému za realizaci nákupu. Objednávka musí obsahovat jednoznačnou identifikaci předmětu nákupu – název, druh, typ, třídu, rozměr, měrnou jednotku, kód, katalogové číslo, apod., pokud to povaha předmětu nákupu vyžaduje. Dalšími povinnými náležitostmi jsou údaje o tom, kdo objednávku vystavil, kontaktní spojení, termín a způsob dodání, razítko a podpis odpovědného zaměstnance požadujícího oddělení.

Zaměstnanec pověřený přezkoumáním a kontrolou ověří úplnost a správnost objednávky předložené zpracovatelem. Je povinen přezkoumat vedle formálních a věcných náležitostí zejména požadavky kvalitativní. V případě zjištění nedostatků vrátí tento zaměstnanec objednávku s připomínkami zpět zpracovateli k dopracování. Objednávku je možné postoupit ke schválení pověřenému zaměstnanci teprve po vyřešení všech nedostatků. Provedené přezkoumání potvrdí pověřený zaměstnanec svým podpisem.

Přezkoumanou objednávku schvaluje svým podpisem zaměstnanec uvedený v čl. 9 směrnice.

Zaměstnanec zodpovědný za realizaci objednávky zajistí odeslání schválené objednávky dodavateli.

2 ANALÝZA POŽADAVKŮ NA INFORMAČNÍ SYSTÉM

Celý proces schvalovacího řízení je v současném řešení prováděn oběhem listinné verze formuláře mezi osobami zúčastněnými v procesu schvalování. Do příslušného formuláře jednotliví zaměstnanci vpisují údaje potřebné v rozhodovacím procesu. Jak již bylo uvedeno, celý postup je determinován pravidly vymezenými ve vnitropodnikové směrnici. Základní požadavky na nový informační systém tak vyplývají z potřeby společnosti přenést klíčové postupy stanoveného procesu do elektronické podoby. To bude ve výsledku znamenat, že do IS bude nutné implementovat systém průchodu požadavku přes taxativně vymezené body. Úlohou IS nicméně bude pokrývat pouze oblast schvalování požadavku. Následný proces realizace nákupu bude již záležitostí interního logistického informačního systému.

Konkrétní požadavky na budovaný informační systém vycházejí z analýzy v prostředí nasazení IS. Nejvýznamnější zdroj, diskuze s účastníky systému, přinesl cenné informace o požadovaném chování IS. Celkový seznam požadavků je uveden v části 4.2 Funkční požadavky, resp. v části 4.3 Nefunkční požadavky.

Z pohledu použitých technologií je požadavek společnosti na použití volně šiřitelného software, který nebude generovat žádné licenční poplatky. Dále má být dostupný neomezenému počtu uživatelů a z pohledu typu koncového zařízení musí být multiplatformní. Tyto požadavky vycházejí do značné míry i z toho, že nemocnice má zkušenosti s využíváním softwarových řešení, které běží pod operačním systémem Linux, přesněji řečeno, jejichž serverová část využívá tohoto prostředí. Z praxe tak může posoudit spolehlivost i nákladovost platformy.

Požadavek na multiplatformnost je dán rozmanitostí koncových zařízení, které se ve společnosti využívají. Nemocnice má instalace různých edic Microsoft Windows, dále Linux, iOS, OS X. S ohledem k tak široké škále variant operačních systémů se pro budovaný informační systém jeví jako nejvýhodnější formát webová aplikace. Tato technologie je napříč všemi vyjmenovanými platformami podporována. Výhodou samozřejmě zůstává také fakt, že v dnešní době široce rozšířeného přístupu k Internetu a k dostupnosti velkého množství webových aplikací, by neměl být pro uživatele zásadní problém rychle se adaptovat na nově vytvořený informační systém.

2.1 Dostupné možnosti řešení

Mezi možné postupy tvorby aplikace se řadí vlastní kompletní vývoj, kdy veškeré kódy napíšu sám. Další variantu představuje použití nějakého stávajícího systému na trhu. Třetí možností je vlastní vývoj s využitím některého z dostupných frameworků.

První varianta kompletního vývoje aplikace by měla význam jedině v případě, že bych jej vytvářel jako znovupoužitelný framework. V současnosti je však na výběr mnoho frameworků, pomocí kterých lze dostatečně řešit zpracovávanou problematiku. Vývoj vlastního řešení by tak byl zbytečný, pracný a časově neúměrně náročný. V konečném důsledku by to vedlo k oddálení termínu nasazení požadovaného informačního systému.

2.1.1 Existující řešení

Pro možnost využití již existujícího softwarového řešení se nabízejí dvě alternativy.

CRM systémy

Jednou ze zkoumaných variant nasazení byly CRM systémy. Tyto systémy se obecně zaměřují na komunikaci se zákazníkem, podporu v ekonomické oblasti, integrují logistické aplikace a systémy pro podporu rozhodování. Po bližším prozkoumání však tyto systémy v mnoha ohledech nevyhovují pro realizaci zamýšleného informačního systému. Zřejmě nejzásadnějším omezením je jejich orientace především na management a řízení vztahů se zákazníkem. Obsahují sice i moduly, které se zaměřují na odbyt, nicméně vše je vztaženo k potřebě získat nebo udržet zákazníka [1]. Z pohledu požadavků na informační systém, by bylo komplikované implementovat do takovéhoto CRM požadované workflow, jež by vyhovovalo současnému podnikovému procesu.

Microsoft SharePoint

Druhou zajímavou alternativou se jevílo využití platformy Microsoft SharePoint. Tento produkt slouží k vytváření webů organizací. Může se používat jako bezpečné místo k ukládání, uspořádání a sdílení informací a přístupu k nim z takřka libovolného zařízení. Mezi dostupné produkty patří SharePoint Online, SharePoint Foundation, SharePoint Server [2]. SharePoint není ovšem pouze knihovnou dokumentů poskytující nad nimi určité množství funkcí. Díky možnostem vytváření pracovních postupů (workflow) získáváme nástroj pro automatizování interních procesů, které doposud probíhaly manuálně. Tvorbu pracovních postupů si můžeme představit jako sadu stavebních bloků, které lze propojovat

dohromady. Lze vytvořit vlastní funkcionalitu, která odpovídá konkrétnímu pracovnímu procesu podniku.

Samotné schéma pracovního procesu je možné připravit v aplikaci Microsoft Visio a přímo jej importovat do prostředí SharePoint Designer ve formátu Visio Workflow Interchange (VWI). Po případné další úpravě se soubor VWI následně importuje na SharePoint server, kde již máme dopředu připravenou datovou vrstvu aplikace [3]. Celý proces tvorby a správy workflow je poměrně jednoduchý a především nabízí vysokou schopnost úprav dle požadavků zákazníka, pro různé podnikové procesy.

Podle typu produktu SharePoint se liší i model licencování. V rámci služby SharePoint Online se licencuje jednotlivým uživatelům. Pro ostatní produkty je potřeba zakoupit licenci pro klientský přístup podle modelu server/CAL. Každá osoba nebo zařízení s přístupem k SharePoint Serveru potřebuje licenci CAL [4].

2.1.2 Framework

Pro usnadnění vývoje aplikace je vhodné využití některého z dostupných frameworků. Jedná se o systémy, které řeší základní problémy a implementují bazální části funkcionality aplikace. Při samotném vývoji se tak lze zaměřit na konkrétní specifika aplikace, bez nutnosti zabezpečovat implementace základních komponent. Mělo by se jednat o stabilní framework, který zajistí podporu aplikace pro případný další vývoj. Framework musí umožnit využití moderních architektur, jako je například architektura MVC³. Samozřejmě by mělo být dobré zabezpečení proti různým typům útoků. Výhodou bude také možnost tvorby znovupoužitelných částí kódu.

Všechny tyto požadavky, které jsou na framework kladeny, splňuje Nette Framework. Tento framework je vhodný i z toho důvodu, že se jedná o jeden z nejpopulárnějších PHP frameworků, který je podporován širokou vývojářskou komunitou [5].

2.1.3 Zhodnocení

S přihlédnutím k požadavkům na informační systém není možné implementovat žádné z existujících řešení. Některé CRM systémy jsou sice poskytovány jako open-source, nicméně obecně svým zaměřením neodpovídají potřebám společnosti v oblasti řešeního

³ MVC – Model-View-Controller

problému. Oproti tomu produkt SharePoint svými možnostmi naplňuje požadavky na výsledný produkt z pohledu funkčních požadavků. Nevýhodou tohoto řešení jsou licenční poplatky, které si zadavatel projektu nepřeje. Je nutné přihlídnout i k tomu, že databáze SharePoint se instaluje pouze do prostředí MS SQL serveru, což může generovat další licenční poplatky.

Společným rysem existujících řešení je poskytování služby ve formátu webové aplikace s vlastním uživatelským rozhraním, které je dostupné přímo ve webovém prohlížeči. Tento koncept, jak jsem naznačil již v analýze požadavků, se jeví jako nejvýhodnější. Dalším důležitým atributem je volba open-source technologií.

Jako výsledné technologické řešení tak budu volit kombinaci webového serveru Apache, skriptovacího jazyka PHP a pro zajištění perzistence dat použiju databázový systém MySQL. Kombinace PHP a MySQL představuje nejčastěji používané prostředky při tvorbě webových aplikací. Pro usnadnění a zrychlení vývoje bude použit Nette Framework.

3 TECHNOLOGIE PRO TVORBU APLIKAČNÍ ČÁSTI

V předchozí kapitole bylo navrženo technologické řešení projektu. Jednotlivé technologie si popíšeme v následujících kapitolách.

3.1 PHP

PHP (PHP: Hypertext Preprocessor) je skriptovací programovací jazyk, určený pro programování dynamických internetových stránek. Nejčastěji se začleňuje přímo do struktury jazyka HTML, XHTML či WML, což lze využít při tvorbě webových aplikací. PHP skripty jsou vykonány na straně serveru a k uživateli je přenášen až výsledek jejich činnosti [6]. PHP je platformě nezávislý a aplikace v něm napsané lze bez větších úprav provozovat na mnoha různých systémech, jako jsou GNU Linux, Microsoft Windows, UNIX. Podporuje nativně nebo přes rozhraní práci se širokou škálou databázových systémů (MySQL, ODBC, Oracle, PostgreSQL, MSSQL) a celou řadu internetových protokolů (HTTP, SMTP, FTP, IMAP, POP3 ...) [6].

PHP je distribuován jako open source projekt. Je tedy dostupný zcela zdarma bez jakýchkoliv licenčních poplatků. Svobodná licence a otevřenost kódu jsou zárukou toho, že případné bezpečnostní chyby budou snáze nalezeny a odstraněny.

3.1.1 Historie

V roce 1995 vytvořil Rasmus Lerdorf sadu skriptů v jazyce Perl pro zpracování informací o přístupech k jeho webu. Tuto sadu nazval „Personal Home Page“ – odtud původní význam zkratky PHP. Později tuto sadu přepsal do jazyka C a rozšířil implementaci o schopnost komunikace s databázemi a tvorbu jednoduchých dynamických aplikací pro web. Výsledný zdrojový kód uvolnil pod označením PHP/FI. V roce 1997 následoval PHP/FI 2.0. Nad existující členskou základnou PHP/FI byl následně společně vybudován nástupce označovaný jako PHP 3.0. Toto byla první verze, která se blížila dnešní podobě PHP. Klíčovými prvky byla podpora objektově orientovaného návrhu a konzistentnější syntaxe jazyka [6].

Nový engine, označovaný jako Zend, představoval další milník ve vývoji PHP. Oficiálně byl uvolněn v roce 2000. Nové jádro přineslo rapidní nárůst výkonu a stalo se základem PHP 4.0. V roce 2004 vychází PHP 5.0 založené na novém jádru Zend Engine 2.0. Obsahuje nový objektový model a představuje tak další vylepšení [6][7].

Podpora jmenných prostorů byla přidána v roce 2009 ve verzi PHP 5.3. To opět přiblížilo PHP k trendu moderních programovacích jazyků.

3.1.2 Výhody

Jazyk PHP se specializuje především na tvorbu webových aplikací. V základní knihovně obsahuje rozsáhlý soubor funkcí, další jsou dostupné v PECL⁴. Nativní podpora mnoha databázových systémů. Díky masové rozšířenosti jazyka je jeho podpora u hostingových společností téměř stoprocentní – PHP je prakticky standardem. Je k dispozici velmi slušná a propracovaná dokumentace. Na webu je možné získat obrovské množství projektů a kódů, které lze zdarma stáhnout, a které řeší obecné i specifické problémy. Distribuován je v režimu velmi svobodné licence což redukuje náklady spojené s provozováním webového sídla [7].

3.2 MySQL

MySQL je multiplatformní databázový systém, kde komunikace s ním probíhá pomocí strukturovaného jazyka SQL. Podobně jako u jiných SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními.

Pro svou snadnou implementovatelnost na různých operačních systémech, pro svůj výkon (od počátku byl optimalizován především na rychlost) a díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na celkovém objemu používaných databází [8].

MySQL podporuje několik typů datových úložišť (storage engine). Každé z nich vyniká v různých typech operací. Vhodným výběrem můžeme dosáhnout navýšení výkonu databázového serveru. Nejpoužívanější jsou úložiště typu InnoDB a MyISAM [9].

Výchozí formát úložiště dat je od verze 5.5 typ InnoDB, který nahradil do té doby používaný formát MyISAM [10].

3.2.1 Formát MyISAM

MyISAM je nejpoužívanějším formátem úložiště až do verze MySQL 5.1. Mezi jeho přednosti patří rychlost čtení. Indexy u těchto typů tabulek jsou komprimovány, zabírají méně místa a systém provede příkazy SELECT na těchto tabulkách s menší spotřebou systémového

⁴ PECL - PHP Extension Community Library

vých zdrojů. Komprimování indexů však má vyšší režii, takže provádění akčních dotazů nad těmito tabulkami spotřebuje více času procesoru. Podporuje fulltextové vyhledávání za pomoci fulltextových indexů, v nichž se indexují jednotlivá slova pro potřeby složitých vyhledávacích operací. MyISAM při databázových operacích uzamyká celé tabulky, nikoliv pouze řádky. Podporuje automatickou i ruční kontrolu a opravu tabulek. MyISAM nepodporuje transakční zpracování. Také nepodporuje cizí klíče [11][12][13].

3.2.2 Formát InnoDB

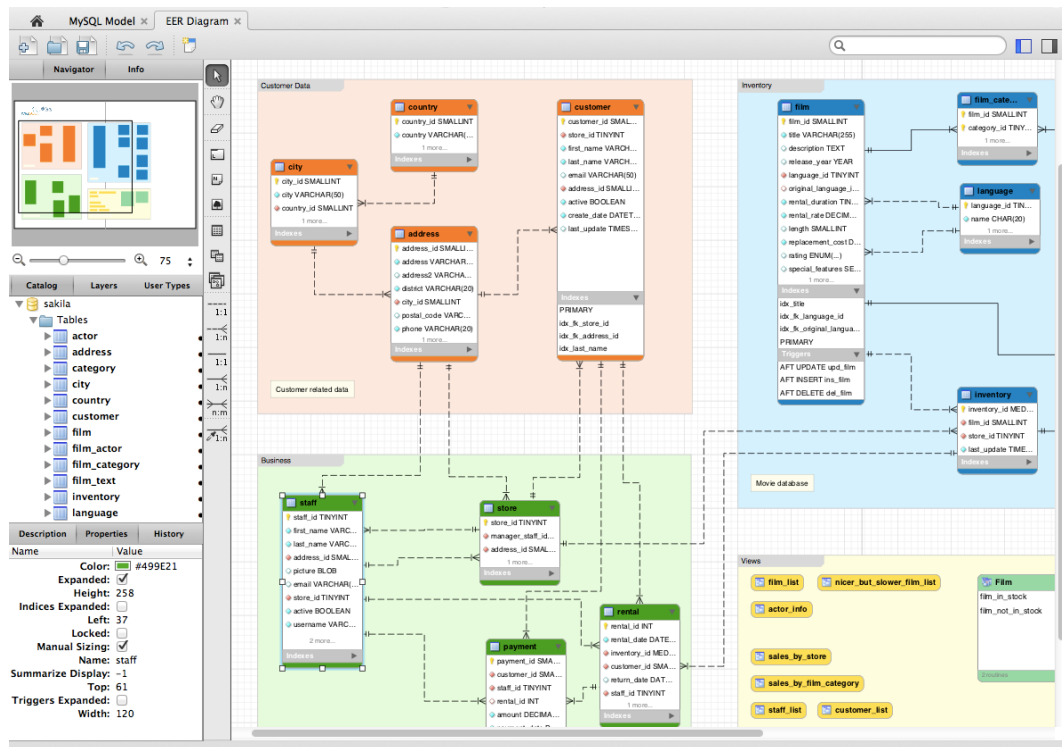
InnoDB je transakční datové úložiště podporující příkazy BEGIN, COMMIT a ROLLBACK. Je optimalizováno pro dotazy typu INSERT a UPDATE a pro práci s velkým objemem dat. Transakční zpracování je klíčové pro spolehlivé dokončení prováděných příkazů. Hodí se tedy pro aplikace, které jsou závislé na naprosto spolehlivé práci s daty, jako jsou například bankovní systémy. Disponuje schopností uzamykání záznamů na úrovni řádků. Mezi jeho další vlastnosti patří podpora cizích klíčů a omezení referenční integrity, body obnovení [14][15].

3.3 MySQL Workbench

MySQL Workbench je open source nástroj od autorů MySQL, který je distribuován pod licencí GPL. Jedná se o multiplatformní program fungující na platformě Linux, Windows, Mac OS X [16]. Možnosti jeho využití se dají rozdělit do oblastí [17]:

- Vizualní modelování struktury databáze (EER modely).
- Automatické generování SQL dotazů.
- Monitorování a analýza chodu databázového serveru.

MySQL Workbench umožňuje operace typu *export*, *import* a *diff*, označované jako *forward engineering* a *reverse engineering*. Je tedy možné z vytvořeného modelu exportovat SQL skript. Případné změny v modelu lze exportovat jako ALTER skript vůči existujícímu SQL skriptu. Podporována je i možnost vytvoření modelu na základě již existující databáze (*reverse engineering*) [16][17].



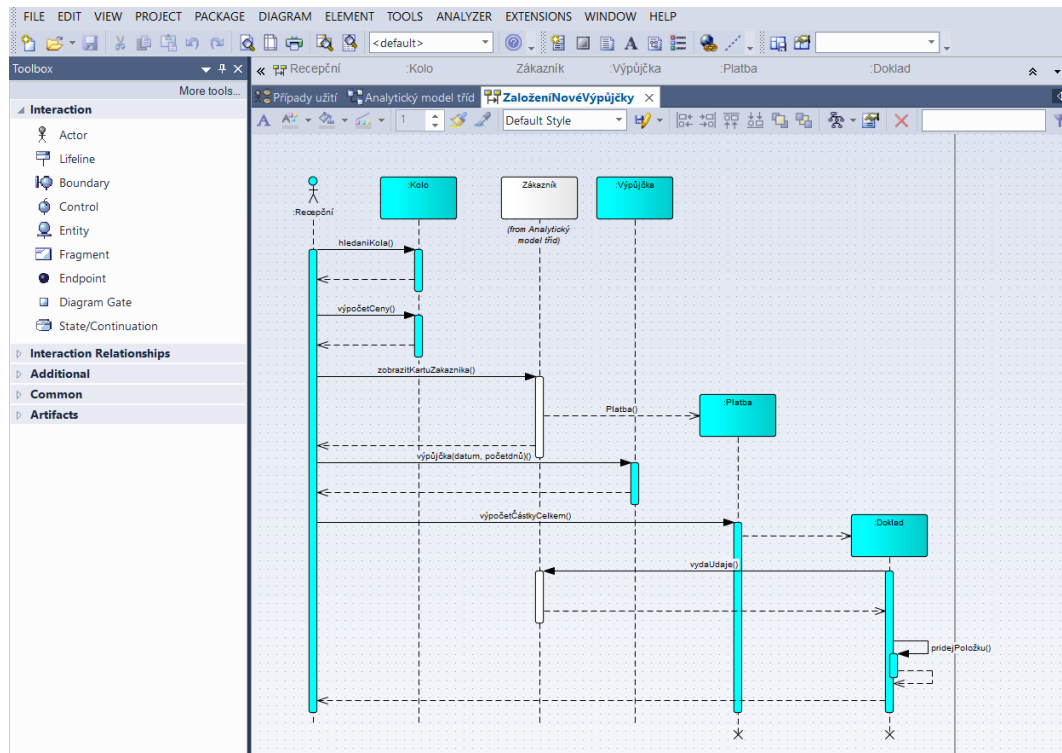
Obr. 2. Aplikace MySQL Workbench [17]

3.4 Enterprise Architect

Enterprise Architect je nástroj používaný pro modelování za pomoci notace UML⁵ jehož autorem je australská společnost Sparx Systems. Jedná se o nástroj, který umožňuje podpořit a usnadnit celou fázi vývoje softwaru, od zadání požadavků, přes analýzu stavů, návrh modelů, až po testování a údržbu. To vše s využitím diagramů v UML. V EA⁶ lze vytvářet širokou škálu diagramů, mezi něž patří Diagram tříd, Diagram komponent, Diagram nasazení, Diagram případů užití, Diagram aktivit, Sekvenční diagram, Business Process Diagram a další [18][19].

⁵ UML – Unified Modeling Language

⁶ EA – Enterprise Architect



Obr. 3. Aplikace Enterprise Architect [19]

3.4.1 UML

Unified Modeling Language je soubor grafických notací, který se používá pro vizualizaci, specifikaci, navrhování a dokumentaci při vývoji softwaru. V oblasti analýzy a návrhu se stal standardem. UML je možné použít třemi způsoby [20]:

- UML jako náčrt – diagramy se ručně kreslí na tabuli nebo do sešitu. Grafická podoba řešeného problému napomáhá jeho lepšímu pochopení. Výhodou diagramů je jejich abstrakce, což umožňuje vyjádřit určitý pohled na systém bez potřeby hlubšího popisu.
- UML jako plán – jedná se o detailnější variantu návrhu než je náčrt. Diagramy jsou vytvářeny za pomoci specializovaných nástrojů a po dokončení slouží dále jako dokumentace.
- UML jako programovací jazyk – z detailního UML diagramu lze vygenerovat šablonu programového kódu, který slouží jako základ pro implementaci.

3.4.2 Business Process Diagram

Business procesy popisují způsob fungování firmy a postupy jak řeší své úkoly. Představují zachycení reality vzájemně propojených aktivit, tvořících společně proces. K jeho popisu

se využívá aktivitních diagramů UML nebo častěji notace Business Process Model Notation (BPMN) [21].

3.4.3 Model případů užití

Use Case Model (Model případů užití) patří mezi dynamické modely, jimiž se popisuje chování systému. Úzce navazuje na požadavky, kdy každý požadavek je realizován případem užití. Při konstrukci modelu je zapotřebí nalézt hranice systému a aktéry případu užití. Mezi zdroje informací, ze kterých vycházíme, patří model business procesů, definice požadavků a doménový model [21].

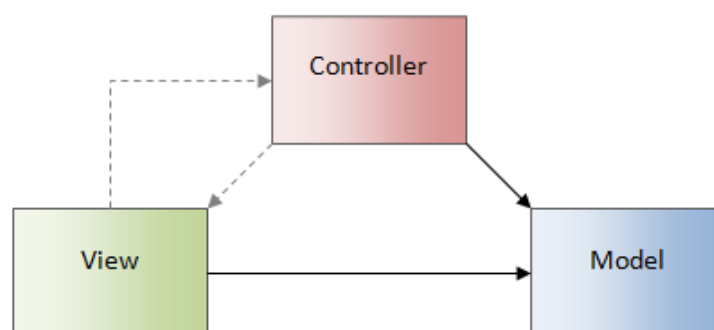
3.5 Architektura MVC

Model-View-Controller je softwarová architektura oddělující kód obsluhy od kódu aplikační logiky a od kódu zobrazujícího data. Jedná se o popis způsobu rozdělení aplikace do logických částí tak, aby je šlo upravovat samostatně a dopad změn byl na ostatní části co nejmenší [22]. Architekturu MVC popsal v roce 1979 Trygve Reenskaug a v dnešní době je tato architektura velmi populární.

Aplikace vytvořená s využitím MVC se skládá ze tří komponent:

- Model,
- View,
- Controller.

Kód obsluhy je reprezentován Controllerem, aplikační logiku představuje Model a kód zobrazující data obstarává View [22].



Obr. 4. Architektura MVC [22]

Rozdělení na logické části se aplikace zpřehledňuje, usnadňuje se budoucí vývoj a umožňuje testování jednotlivých částí zvlášť [23].

3.5.1 Model

Vrstva Model zahrnuje data a aplikační logiku. Nejedná se však o pouhý ovladač pro manipulaci s daty, ale o jejich reprezentaci. Jakákoliv akce uživatele představuje akci Modelu. Model spravuje svůj vnitřní stav a navenek nabízí pevně dané rozhraní [23]. Pro tuto vrstvu je typické, že nemá žádnou vazbu na View nebo Controller. Úkolem Modelu je poskytnout abstrakci datům, se kterými aplikace pracuje. Z pohledu ostatních vrstev přitom vůbec nezáleží, jakým způsobem jsou data uložena, zda perzistentně či v nějaké variantě kombinovaného úložiště, přistupují k nim pomocí rozhraní, které Model nabízí [22].

3.5.2 View

View je vrstva aplikace, která má za úkol nějakým vhodným způsobem zobrazit data získaná z modelové vrstvy. U webových aplikací je takovým výstupem HTML kód, obrázek, multimediální obsah, apod. Tato vrstva obvykle používá šablonovací systém, který popisuje, jak se má výsledek zobrazit [23]. Vrstva Model mívá na schématech zakreslenu nepřímou vazbu na View. Tím se ukazuje na skutečnost, že pokud se data Modelu změní, potom je View upozorněn nějakým notifikačním mechanismem a dochází k novému vykreslení dat [22].

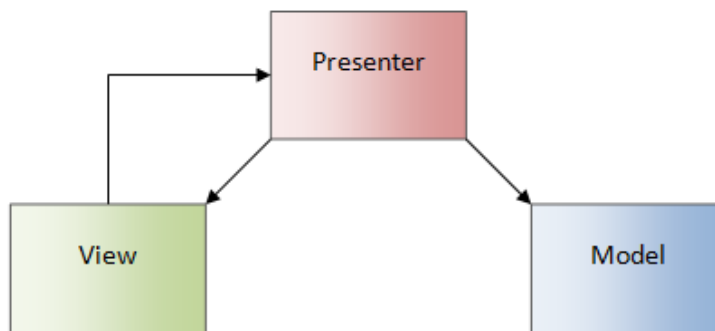
3.5.3 Controller

Controller je řadič, který zpracovává požadavky uživatele. Představuje vstupní bod aplikace, jehož úkolem je na základě vstupní akce zajistit požadovanou odezvu. Úkolem Controlleru je zavolat příslušnou aplikační logiku, která uloží nový stav do Modelu a následně zajistí zavolání správného View [23].

3.6 Architektura MVP

Model-View-Presenter je speciální typ MVC architektury. Vzor MVC vznikl historicky jako obraz tehdejšího stavu hardwaru, kdy vstup a výstup byly na sobě do značné míry nezávislé. View vykresloval uživatelské rozhraní a Controller zpracovával vstup. Postupem času, tak jak se vyvíjely operační systémy a programovací jazyky, se input a output spojily do jednoho. Např. Button zvládá jak své vykreslení tak ošetření události vstupu klávesnice, myši apod. Do značné míry tak zanikla potřeba ošetřovat vstup za pomoci Controlleru [24].

S nástupem webu v devadesátých letech minulého století se ovšem na světě opět objevuje platforma, která neumí sloučit vstup (URL) a výstup (HTML). Vzniká tak nový architektonický model MVP. Rozdíl proti MVC je ve způsobu implementace vrstvy Controller, kterou zde v tomto případě zastupuje Presenter. Ten typicky vystupuje jako prostředník mezi Modelem a vrstvou View. Presenter manipuluje s Modelem a pomocí systému notifikací aktualizuje View, nebo View ovlivňuje přímo [24].



Obr. 5. Architektura MVP – vzor Passive View [24]

3.7 Nette Framework

Nette Framework je open source Framework pro tvorbu webových aplikací, který využívá návrhového vzoru MVP [25]. Zaměřuje se na eliminaci bezpečnostních rizik, jako jsou XSS, CSRF, session hijacking, session fixation [27]. Podporuje technologie AJAX, Dependency Injection, SEO, DRY, KISS, atd. [28].

Mezi další přednosti patří [26][28][29]:

- silný validační jazyk,
- podpora automatického překladu,
- událostmi řízený běh aplikace,
- dokumentace a podpora komunity,
- vyzrálý a čistý objektový návrh,
- open-source licence New BSD nebo GNU General Public License (GPL) ve verzi 2 nebo 3.

3.7.1 Dependency Injection

Součástí platformy Nette Framework je pokročilý návrhový vzor Dependency Injection. Jeho úlohou je řešit závislosti mezi třídami a převzít na sebe odpovědnost za získávání objektů, které potřebují ke své činnosti. K tomu využívá koncept systémového kontejneru, v němž jsou definovány všechny služby a parametry potřebné pro běh aplikace [30]. Objekty služeb jsou vytvářeny pouze prostřednictvím kontejneru. Implementace tohoto návrhového vzoru je zajišťována třídou `Nette\DI\Container`, která automaticky zabezpečuje, že z každé služby se vytvoří pouze jedna instance. Platí také, že služby jsou vytvářeny až v případě potřeby [31]. K jejich vytvoření vůbec nedojde, pokud by se např. na dané stránce vůbec nepoužily.

Na platformě Nette Framework jsou k dispozici čtyři hlavní způsoby, jak do aplikačních tříd předávat závislosti [32]:

- předávání konstruktorem,
- předávání setterem nebo nastavením členské proměnné,
- metodou `inject*`,
- anotací `@inject` u proměnné s typem přístupu `public`.

Předávání závislostí metodou `inject*` je způsob specifický pro DI Container v Nette Framework. Jedná se o speciální případ setteru, metoda začíná prefixem `inject`. Předpokládejme, že v systémovém DI kontejneru jsou třídy `MyService` a `AnotherService` registrovány jako služba. Můžeme následně použít tuto konstrukci [32]:

```
1 class MyService
2 {
3     /** @var AnotherService */
4     public $anotherService;
5
6     public function injectAnotherService(AnotherService $service)
7     {
8         $this->anotherService = $service;
9     }
10 }
```

Dalším způsobem, specifickým pro DI Container v Nette, je použití anotace `@inject`, která se uvádí v dokumentačním komentáři. Pro tytéž třídy jako na příkladu shora, definované v systémovém kontejneru, vypadá konstrukce následovně [32]:

```

1 class MyService
2 {
3     /** @inject @var \App\AnotherService */
4     public $anotherService;
5 }

```

Uvedené návrhové způsoby poskytují výhodu přehlednosti a čistoty zdrojového kódu, ve kterém jsou jednotlivé závislosti přehledně viditelné. Kód se zbytečně nekomplikuje dodatečnými konstrukcemi [32].

Předávání závislostí pomocí konstruktoru je dostupné u všech vytvářených tříd a použití setteru u všech nepovinných závislostí. Metody `inject*` a `@inject` jsou dostupné pouze v presenterech [32]. Doporučené způsoby předávání závislostí jsou uvedeny na obrázku (Obr. 6).

Co kde použít?	Presenter	Služba	Komponenta s továrnou
Konstruktor	★★★	★★★★	★★★★
Metoda <code>inject</code>	★★★	★	★
<code>@inject</code>	★★★★	⊗	⊗

★★★ - doporučeno
 ★★ - klidně použijte
 ★ - vyvarujte se
 ⊗ - vážně nedoporučeno

Obr. 6. Způsoby předávání závislostí [33]

3.7.2 Vrstva Presenter

Jak již bylo uvedeno, platforma Nette Framework vychází z architektury MVP. V této architektuře jsou obdobou kontrolerů presentery. Vrstva Presenter zprostředkovává komunikaci mezi vrstvou Model a View. Jinak řešeno, presenter přijímá požadavek a zajistí pro něj odpověď. V rámci objektového přístupu by každá funkčně odlišná část aplikace měla mít vlastní třídu. Tato třída pak bude potomkem třídy `Nette\Application\UI\Presenter`. Každá komponenta vrstvy Presenter má jednu nebo více akcí, o kterých bude řeč dále [23].

3.7.3 Zpracování akce presenteru

Aplikace je spuštěna souborem `index.php`, který se nachází zpravidla v kořenovém adresáři webu. Každý požadavek na aplikaci je posílán prohlížečem právě přes tento soubor. Samotný požadavek však není souborem `index.php` zpracován, ten dále předává řízení do

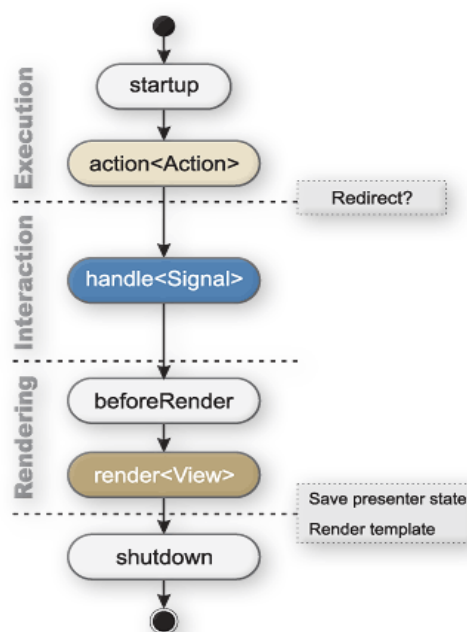
aplikace zaváděcímu souboru *bootstrap.php*. Soubor *bootstrap.php* zajistí ve třídě `Nette\Configurator` vytvoření systémového kontejneru (DI Container). Vytvořený kontejner se uloží do lokální proměnné a následně je volána metoda startující vlastní aplikaci. Všechny HTTP požadavky se dostanou přes soubory *index.php* a *bootstrap.php* do objektu `$application`, ze kterého jsou pomocí routeru překládány na parametry identifikující konkrétní presenter a akci v rámci tohoto presenteru, která se má vykonat [23].

3.7.4 Životní cyklus presenteru

Akce presenteru voláme pomocí konstrukce `metoda<Akce>`. V rámci presenterů můžeme vytvořit (případně překrýt) následující předdefinované metody, které jsou prováděny sekvencně v následujícím pořadí [23]:

- `startup`,
- `action`,
- `handle`,
- `beforeRender`,
- `render`,
- `shutdown`.

Životní cyklus presenteru je možné kdykoliv v průběhu zpracování přerušit zavoláním speciálních metod `terminate`, `sendTemplate`, `sendPayload`, `sendResponse`. Lze jej také ukončit přesměrováním nebo vyvoláním výjimky `BadRequestException` [23].



Obr. 7. Životní cyklus presenteru [23]

Jednotlivé metody mají tento význam [23]:

- Metoda `startup` – volá se po vytvoření presenteru a je společná pro všechny jeho akce. Pomocí této metody se inicializují proměnné. Používá se také k ověření uživatelských oprávnění. Tuto metodu lze přerušit.
- Metoda `action*` – tato metoda je unikátní pro každou akci. Pokud máme například akci `view`, bude mít příslušná metoda název `actionView`. Tato metoda se volá dříve než metoda `render`. Využívá se pro ty situace, kdy presenter provede určitý úkon, např. přihlásí uživatele, a poté přesměruje jinam.
- Metoda `handle*` – využívá se pro zpracování signálů. Používají ji např. formuláře. Určeno zejména pro komponenty a zpracování AJAXových požadavků.
- Metoda `beforeRender` – metoda je určena k překrytí, obdobně jako `startup`. Provádí se před vykreslením šablony a slouží k jejímu nastavení, předání proměnných společných pro všechny akce.
- Metoda `render*` – slouží k naplnění šablony požadovanými daty. Pro akci `edit` se bude tato metoda jmenovat `renderEdit`.
- Metoda `shutdown` – je volána těsně před ukončením životního cyklu presenteru. Metodu je možné přerušit.

3.7.5 Přístup k databázi

Nette Framework poskytuje vestavěnou vrstvu pro práci s databázemi nazvanou Nette Database. Tato vrstva podporuje přístup k databázím MySQL, PostgreSQL, Sqlite, Oracle, MS SQL, ODBC⁷. Základní třídou je `Nette\Database\Connection`, která tvoří obálku nad PDO⁸ a reprezentuje připojení k databázi [34]. PDO vytváří jednoduchou abstraktní vrstvu pro přístup k databázi, poskytující výhody ve formě konzistentního kódování, flexibility, objektově orientovaného přístupu a výkonu. Pro různé typy databází se vždy používají stejné funkce pro zadávání příkazů a získávání dat z databáze [36].

⁷ ODBC – Open Database Connectivity. Jedná se o rozhraní společnosti Microsoft pro přístup k datům v prostředí relačních a non-relačních databází [35]

⁸ PDO – PHP Data Objects

Pro vytvoření připojení k databázi stačí vytvořit novou instanci třídy `Nette\Database\Connection`:

```
1 use Nette\Database\Connection;
2 $connection = new Connection($dsn, $user, $password);
```

Všechna připojení k databázi se navazují až ve chvíli, kdy je to poprvé potřeba. Nikoliv tedy v okamžiku vzniku instance `Connection`. Tento způsob připojení je označován jako „líný“ [34]. Vhodným způsobem, jak nastavit přístup k databázi, je zahrnutí definice do konfigurace aplikace, odkud je dále poskytován přes systémový kontejner.

Základní funkcionalitu poskytuje třída `Nette\Database\Context`, která umožňuje klást databázové dotazy voláním metody `query` [34]. Tato třída vychází z knihovny `NotORM`⁹. Základním rysem její práce je efektivnost při získávání dat. Klade jen minimální množství jednoduchých dotazů a z databáze přenáší jen ta data, která jsou potřeba. Pro plné využití síly této knihovny jsou zapotřebí definované cizí klíče [37][38].

Třída `Nette\Database\Table` poskytuje pokročilou vrstvu pro práci s tabulkami. Její úlohou je zjednodušovat a optimalizovat výběr dat z databáze. Jedním z principů je poskytnout programátorovi takové rozhraní, aby nemusel řešit, jakým způsobem se provede výběr dat z databáze, ale mohl psát pouze podmínky. `Nette\Database\Table` tak, díky vlastnosti cizích klíčů, samostatně spravuje vazby mezi tabulkami. Tuto schopnost pak využívá vnitřní parser při sestavování SQL dotazů. S vytvořenými vazbami je pak možné pracovat dvěma způsoby [39]:

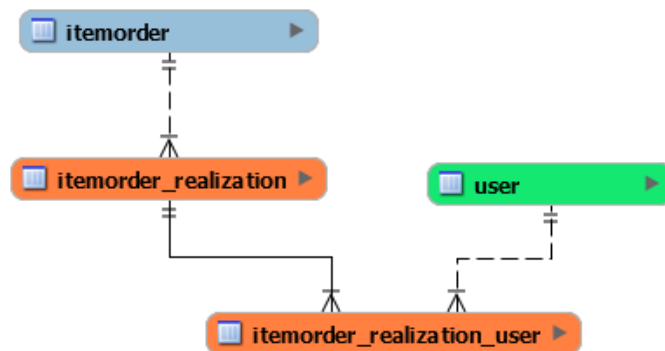
- filtrovat záznamy, které vrací třída `Nette\Database\Table\Selection`,
- získat data z vybraného `ActiveRow`¹⁰.

Pro práci s konkrétní tabulkou je potřeba nejprve vytvořit novou instanci objektu `Nette\Database\Table\Selection`. To se provede zavoláním metody `table` na objektu `Nette\Database\Table\Context`. Jako argument se uvede název požadované tabulky. Nad objektem `Table\Selection` je možné volat metody pro výběr, vkládání, aktualizaci či mazání záznamů [40].

⁹ NotORM – PHP knihovna pro jednoduchou práci s daty v databázi

¹⁰ ActiveRow – instance třídy `Nette\Database\Table\ActiveRow`

Logika vytvoření spojovacího klíče mezi tabulkami je dána implementací `IConventions`¹¹, která analyzuje cizí klíče a umožňuje pracovat se vztahy mezi tabulkami. K tomu se používá tzv. tečková, resp. dvojtečková notace. Tečková notace se používá pro přístup k tabulce ve vztahu 1:1, resp. 1:N. Pro vytvoření spojení N:1 použijeme dvojtečkovou notaci [40].



Obr. 8. Schéma databáze

Uvažujme databázi dle uvedeného obrázku (Obr. 8). Následujícím způsobem získáme výpis všech záznamů z tabulky `itemorder_realization`, kde přidělený řešitel má jméno „Petr“.

```
1 $selection = $connection->table('itemorder_realization')
2   ->where(':itemorder_realization_user.user.firstname ?',
3         'Petr')
4   );
```

Níže uvedeným způsobem pak získáme výpis všech záznamů z tabulky `itemorder_realization`, kde název realizované položky obsahuje řetězec „Počítač“.

```
1 $selection = $connection->table('itemorder_realization')
2   ->where('itemorder.name LIKE ?', 'Počítač%');
```

Jednotlivé řádky, které vrací `Table\Selection` jsou reprezentovány instancí třídy `Nette\Database\Table\ActiveRow`. Tato třída poskytuje standardní rozhraní pro přístup k jednotlivým sloupcům, a to buď formou čtení dat jako členské proměnné nebo lze použít přístup přes klíče pole [41].

3.7.6 Šablonovací systém Latte

Šablony a šablonovací systémy slouží k oddělení aplikační logiky (backend) od prezentační části webu (frontend). Obě části se zpřehlední, zjednoduší se údržba stránek a lze tímto

¹¹ `IConventions` – rozhraní `Nette\Database\IConventions`

způsobem dosáhnout efektivního a znovupoužitelného kódu. Další výhodou může být rozdělení práce na aplikaci mezi grafika a programátora. Šablonu si můžeme představit jako dokument, jež obsahuje jednak statická data, ale navíc i zástupné symboly, jako jsou odkazy na proměnné, podmínky, cykly a volání funkcí. Šablonovací systém má za úkol takovou šablonu zpracovat a vrátit výsledný dokument. Součástí Nette Framework je šablonovací systém Latte [42].

Latte disponuje sadou vestavěných maker a filtrů. Nabízí také možnost tvorby vlastních maker a filtrů, které se dají do Latte jednoduše zaregistrovat. Systém pracuje se soubory typu *.latte*, ve kterých je možné používat dědičnost. Soubory se také dají rozdělit do bloků, přičemž potomci mohou tyto bloky upravovat, doplňovat apod. Díky tomu je možné, aby společný vzhled stránek byl v jednom souboru, který bude obsahovat vyznačené bloky pro proměnlivý obsah. Nette pro tento účel nabízí šablonu pojmenovanou *@layout.latte*, ale lze pochopitelně vytvořit jakoukoliv jinou. V souboru šablony je možné kombinovat standardní HTML, včetně vloženého kódu jazyka JavaScript, již zmiňovaných maker, filtrů, bloků, atd [42].

Příkladem makra můžou být makra *n:if*, *n:foreach*. Makro *n:if* nejprve ověří existenci proměnné `$items` a následně jej iteruje v makru *n:foreach*, kdy každý iterovaný prvek bude obsažen v HTML elementu `li` [42].

```
1 <ul n:if="$items">
2     <li n:foreach="$items as $item">{$item|capitalize}</li>
3 </ul>
```

Za zmínku stojí použití filtru `capitalize`, který převede první písmeno každého slova na velké [42].

3.7.7 Ochrana escapováním

Důležitou vlastností Latte je, že vypisované proměnné automaticky escapuje. Tedy zajišťuje, že znaky se speciálním významem v HTML budou nahrazeny za jiné odpovídající sekvence. Všechny řídicí znaky HTML kódu jsou překódovány do jejich bezpečné alternativy. Díky tomu je zajištěna ochrana proti útoku typu Cross Site Scripting (XSS). Toto výchozí chování je možné potlačit přidáním filtru `noescape` u konkrétní proměnné [42].

3.7.8 Konfigurace

K zápisu konfigurace v Nette Framework se používá NEON soubor, který je umístěn obvykle ve složce *config* adresářové struktury aplikace. Název souboru je *config.neon*. Lze však mít zapsáno i více konfiguračních souborů, které se k hlavní konfiguraci připojují v sekci *includes*. V části věnované tvorbě systémového kontejneru bylo uvedeno, že tento kontejner obsahuje všechny služby a parametry potřebné pro běh aplikace. Definice těchto objektů je možné provést právě pomocí konfiguračního souboru aplikace. Tento soubor obsahuje různé, tematicky oddělené sekce. Můžeme zde uvést například konfiguraci připojení k databázovému serveru – sekce *database*, definici mailového serveru – sekce *mailer*, parametry Nette Framework – např. sekce *session*, atd. Do sekce *services* se pak umísťují definice jednotlivých služeb [30].

3.7.9 Formuláře

Platforma Nette Framework nabízí propracovaný systém tvorby formulářů, které jsou instancí třídy `Nette\Forms\Form`. Nejen samotný formulář, ale i jeho jednotlivé formulářové prvky jsou reprezentovány jako objekty. Těmto objektům je možné pomocí jejich vlastností nastavit hodnoty, např. podmínky, výchozí hodnoty, apod. Formuláře se automaticky validují na straně klienta i na straně serveru. Na straně klienta se pro kontrolu položek používá jazyk JavaScript. K tomu je nutné do šablony zalinkovat soubor *netteForms.js*. Tento typ kontroly se provádí na úrovni webového prohlížeče. Teprve po úspěšné kontrole na straně klienta jsou data zaslána na server. Odeslaný formulář je interpretovaný jako objekt, který obsahuje zadané hodnoty. K získání těchto hodnot můžeme použít funkci `getValues` [43].

```
1 $values = $form->getValues();
```

U formulářů v presenterech se místo třídy `Nette\Forms\Form` používá od ní odvozená třída `Nette\Application\UI\Form`.

Příklad použití:

```
1 $form = new UI\Form;
2 $form->addText('name', 'Jméno:');
3 $form->addPassword('password', 'Heslo:');
4   ->setRequired('Vyplňte heslo');
5   ->addRule(Form::MIN_LENGTH, 'Min. délka hesla jsou %d znaky', 3);
6 $form->addSubmit('login', 'Registrovat');
```

```
7 $form->onSuccess[] = function ($form, $values) {
8     // zde zpracování formuláře
9 };
10 return $form;
```

Součástí řešení webových formulářů je i zabezpečení aplikace před útoky XSS, CSRF, odfiltrování kontrolních znaků ze vstupů, ověření validity UTF-8 kódování, kontrola na podvržení položek v select boxech atd. [43]

Ochrana proti útoku Cross-Site Request Forgery (CSRF) spočívá v generování autorizačního tokenu a jeho zpětnému ověřování. Ochránit formulář před CSRF lze v Nette Framework zavoláním metody [27]:

```
1 $form->addProtection();
```

Použití této ochrany je doporučeno zejména pro administrační část webu [27].

3.8 Technologie pro tvorbu Frontend části

Pro výstup webové aplikace jsou používány zejména technologie HTML - pro strukturu a textový obsah, a CSS společně s obrázky, které tvoří grafickou podobu webu. Často jsou doplněny dalšími technologiemi, které zajišťují interaktivitu stránek, jako např. JavaScript, Java applety, Flash apod. Webový prohlížeč výsledný kód zpracuje a zobrazí uživateli. Dnes již běžným požadavkem na výstup je jeho schopnost přizpůsobení se a správné interpretace na různých typech koncových zařízení.

K vyřešení zobrazování napříč různě velkými displeji jsou v současné době k dispozici tzv. frontend frameworky. Mezi moderní framework pro oblast HTML a CSS se řadí framework Twitter Bootstrap, pro JavaScript je to pak framework jQuery a jQueryUI.

3.8.1 Twitter Bootstrap

Twitter Bootstrap nabízí sadu nástrojů pro vytváření moderních webových aplikací. Podporuje nejrozličnější webové technologie (HTML, CSS, JS) a mnoho prvků, které se dají snadno implementovat [44].

Byl vytvořen vývojáři Twitteru jako požadavek na tvorbu univerzálního, znovupoužitelného rozhraní. V roce 2011 byl Twitter Bootstrap vydán jako open-source [44]. Velmi brzy po svém uvolnění se stal nejoblíbenějším frameworkem pro tvorbu frontend části webových aplikací. Mezi jeho přednosti patří [45]:

- široká paleta komponent uživatelského rozhraní,

- responzivnost,
- podpora starších prohlížečů,
- rychlost učení.

3.8.2 Knihovny jQuery a jQueryUI

Knihovny jQuery a jQueryUI patří mezi javascriptové frameworky zaměřené na uživatelské rozhraní. Umožňují vytvářet dynamické webové aplikace s rozhraním srovnatelným s desktopovými aplikacemi. Knihovny jsou vyvíjeny pod licencí MIT a GPL a je možné je bezplatně použít [46][47].

jQuery disponuje funkcemi pro obsluhu událostí, manipulaci s CSS, AJAX, výběr DOM elementů, efekty a animace, atd [46].

jQueryUI obsahuje velké množství pluginů, které usnadňují vývoj aplikace. Zaměřuje se také na vylepšení funkcionality HTML prvků, což umožňuje výrazné vylepšení uživatelského rozhraní aplikace. Framework se skládá ze čtyř částí [48]:

- Interactions – metody interakcí mezi uživatelem a rozhraním.
- Widgets – elementy uživatelského rozhraní.
- Effects – grafické efekty.
- Utilities – nástroje pozicování UI komponent, tvorba vlastních widgetů.

II. PRAKTICKÁ ČÁST

4 ANALÝZA A NÁVRH ŘEŠENÍ

Informační systém je určen pro řízení schvalovacího procesu objednávek služeb a zboží. Aby bylo možné definovat požadavky na systém, je potřeba nejprve popsat způsob řešení tohoto úkolu v rámci nastavených procesů nemocnice. V této fázi budu vycházet z platné podnikové směrnice o nakupování, kde jsou určeny jednotlivé etapy schvalovacího procesu a kde jsou definováni příslušní aktéři. Vhodným prvotním krokem analýzy bude tzv. modelování procesu podniku. Smyslem tohoto modelování je systematicky nalézt případy užití. Zvolený postup odpovídá variantě vyhledávání případů užití na základě popisu procesů, která se v posledních letech dostává do popředí před tzv. vyhledáváním na základě znalosti aktérů [49]. Z takto zachycené reality následně vymezím přehled požadavků, které budou popisovat, co musí aplikace splňovat. Tyto požadavky budou rozděleny do dvou skupin, a to do modelu funkčních požadavků a modelu nefunkčních požadavků. Dalším krokem bude sestavení diagramů případů užití, kterými popíšu očekávanou funkcionalitu systému a rozdělím tak aplikaci na menší logické celky. V následném kroku pak tyto analýzy využiji pro konceptuální datové modelování. Budu tak schopen navrhnout model databáze.

4.1 Modelování podnikového procesu

Business Process bude zachycen nejprve v podobě hlavního toku událostí, jenž bude popisovat přímý průběh schvalovacího procesu od založení objednávky až po její realizaci. Následně bude popsán průběh podnikového procesu na úrovni atomických aktivit. Tento způsob již bude obsahovat průběh schvalovacího procesu včetně alternativních toků.

4.1.1 Hlavní tok

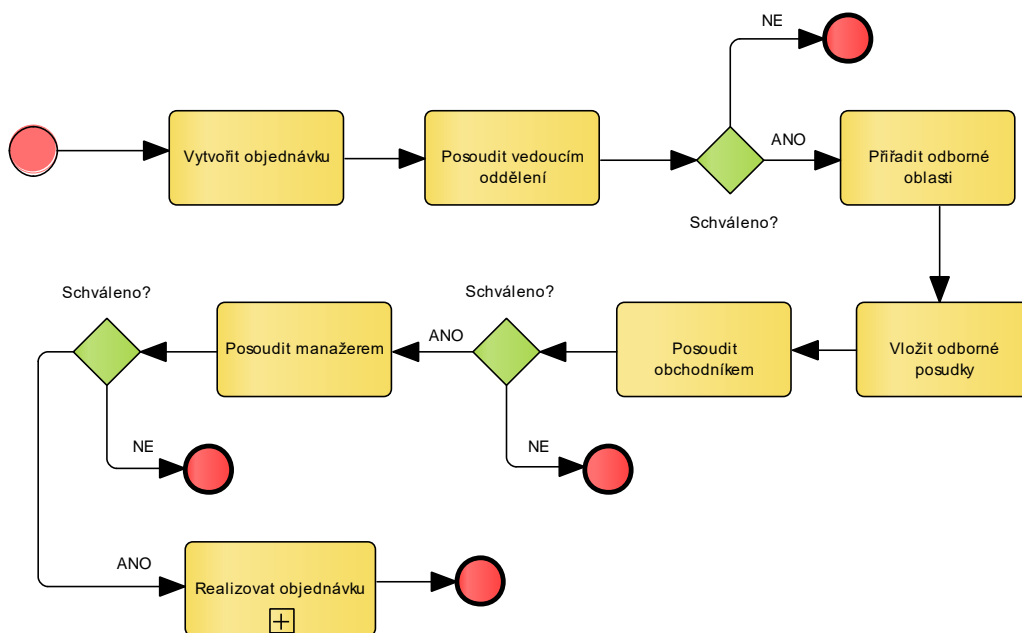
Proces schvalování objednávky lze zachytit jako stav, kdy objednávka prochází sekvenčním způsobem cestou jednotlivých zaměstnanců, kteří se k ní vyjadřují. Na obrázku (Obr. 9) je znázorněn hlavní tok procesu.

Celý průběh začíná vytvořením objednávky. Tuto kompetenci má pověřený pracovník požadujícího oddělení a z ekonomického pohledu (přičtení nákladů) musí být žadatelem konkrétní nákladové středisko. To vyplývá z požadavku adresně zaúčtovat náklady konkrétnímu nákladovému středisku po ukončení realizace. Objednávku následně posuzuje vedoucí příslušného požadujícího nákladového střediska. V případě, že objednávku neschválí, proces schvalování je ukončen. Pokud objednávku schválí, je dále předána zaměstnanci oddělení centrálního nákupu, který určí, v jaké odborné oblasti je vyžadováno

objednávku posoudit. Na základě povahy věci (zboží, služba, prostředek) přiřadí požadavku jednu či více odborných oblastí. Objednávka je v dalším kroku předána zaměstnanci (zaměstnancům), odpovědnému (odpovědným) provést odborný posudek v přidělené oblasti. Pokud je objednávka zařazena do více oblastí, pak se k ní musí povinně vyjádřit všichni experti tak, aby nechyběl žádný z vyžádaných posudků. Objednávka jinak nemůže být postoupena dále. Máme-li všechny vyžádané odborné posudky vypracovány, je objednávka v dalším kroku posuzována zaměstnancem obchodního oddělení. Ten objednávku posoudí jak z formální, tak věcné stránky. Má právo objednávku schválit, zamítnout, případně vrátit k dopracování. Vrácení k dopracování je alternativní krok, který není součástí hlavního toku. Bude popsán později.

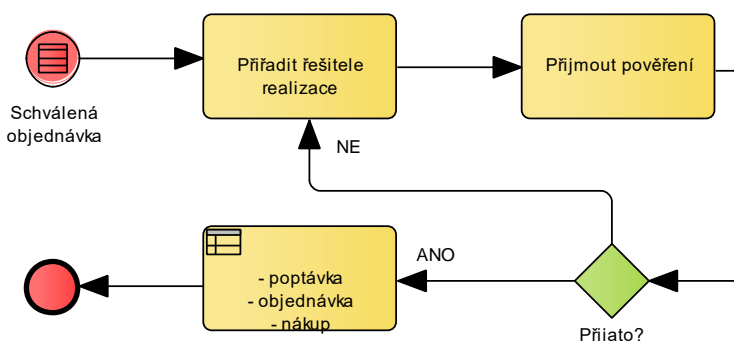
Zamítnutí objednávky obchodníkem ukončuje proces schvalování. Její schválení posouvá objednávku k dalšímu aktérovi, kterým je manažer pro hospodářskou a technickou správu nemocnice. Manažer má právo objednávku zamítnout a proces schvalování ukončit s negativním výsledkem. Může objednávku schválit a ta se dále posouvá k realizaci. Touto aktivitou hlavní tok schvalovacího procesu končí. Realizace objednávky je zastoupena samostatným procesem, který je uveden na obrázku (Obr. 10).

Alternativní variantou, kterou může manažer provést, je vrácení objednávky obchodníkovi k dopracování. Tento alternativní scénář bude součástí podrobného diagramu business procesu.



Obr. 9. Diagram hlavního toku procesu schvalování objednávky

Po schválení objednávky je spuštěn proces realizace. Je provedena poptávka u dodavatelů, vyhodnocení nabídky, objednání u dodavatele a nákup. K provedení těchto kroků je pověřen zaměstnanec, zodpovědný za realizaci objednávky podle předmětu nákupu. Pokud však vyhodnotí, že mu realizace nepřísluší, může pověření odmítnout. Zaměstnanec obchodního oddělení, zodpovědný za řízení procesu schvalování a realizace objednávky, v takovém případě pověří realizací jinou osobu. Do doby převzetí pověření může řešitele měnit i ze své vlastní vůle. To je výhodné v případě, že zvolený řešitel je např. zaneprázdněn jinými realizacemi a naproti tomu jeho kolega se řešené realizace může ujmout díky své volné kapacitě.



Obr. 10. Diagram hlavního toku procesu realizace schválené objednávky

4.1.2 Diagram atomických aktivit

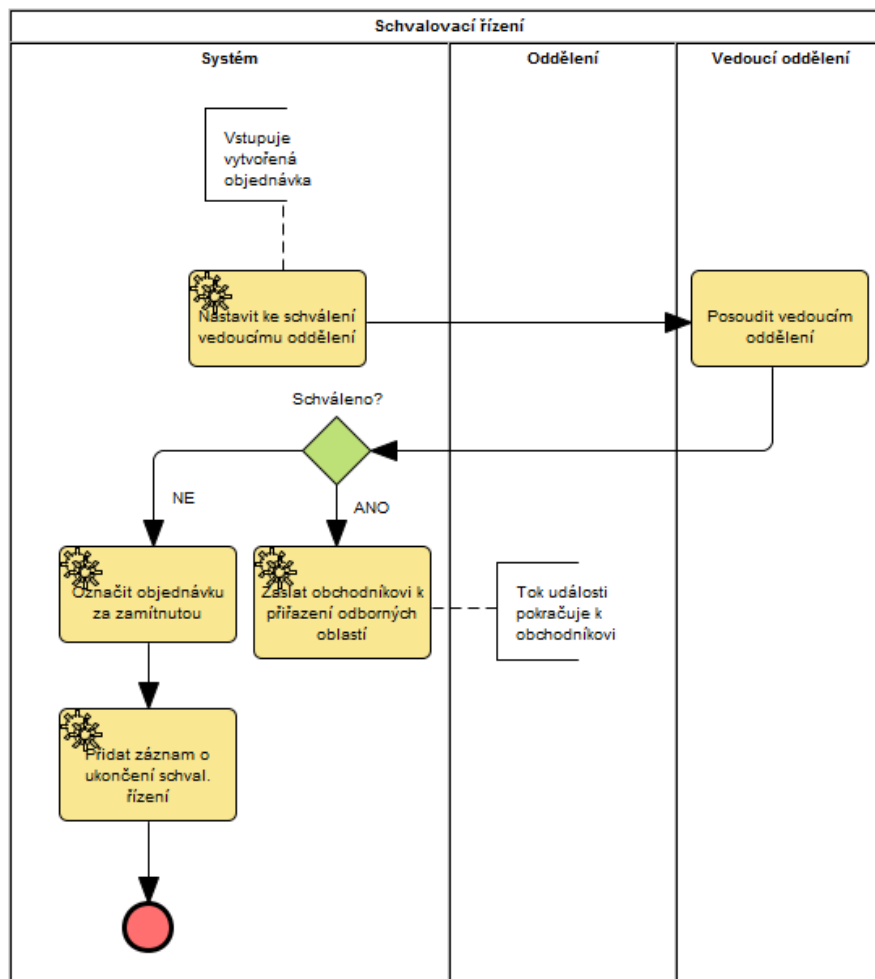
Z hlavního toku schvalovacího procesu můžeme vyjmenovat funkce, resp. role, které vystupují v pozicích hlavních aktérů. Na základě této skutečnosti postoupíme v modelování business process diagramu na úroveň atomických aktivit.

Celý proces schvalování objednávky lze uzavřít do jednoho poolu (bazénu), kde jednotlivé dráhy budou tvořeny shora vymezenými funkcemi (rolemi). V příloze (Příloha P III) je uveden kompletní diagram, včetně průběhu alternativních rozhodnutí, které nebyly součástí popisu hlavního toku. V následující části se zaměřím jen na ty aktivity a sekvenční toky, které jsou rozbočeny nebo slučovány bránou. Bude se tedy jednat o „výstřižky“ z celkového schématu.

Rozděleno podle funkcí budou aktéry *Oddělení*, *Vedoucí oddělení*, *Expert*, *Obchodník*, *Manažer*. Součástí bazénu budou také aktivity, které budou delegovány na informační sys-

tém. Jedna z drah v bazénu tak bude představovat aktéra *System*. Podívejme se nejprve na možnosti vedoucího oddělení.

Po vytvoření objednávky *System* nastaví požadavku příznak ke schválení a objednávka je doručena ke stanovisku *Vedoucímu oddělení*. Podle rozhodnutí vedoucího oddělení dostává objednávka příznak ukončení nebo příznak zaslání obchodníkovi (Obr. 11).

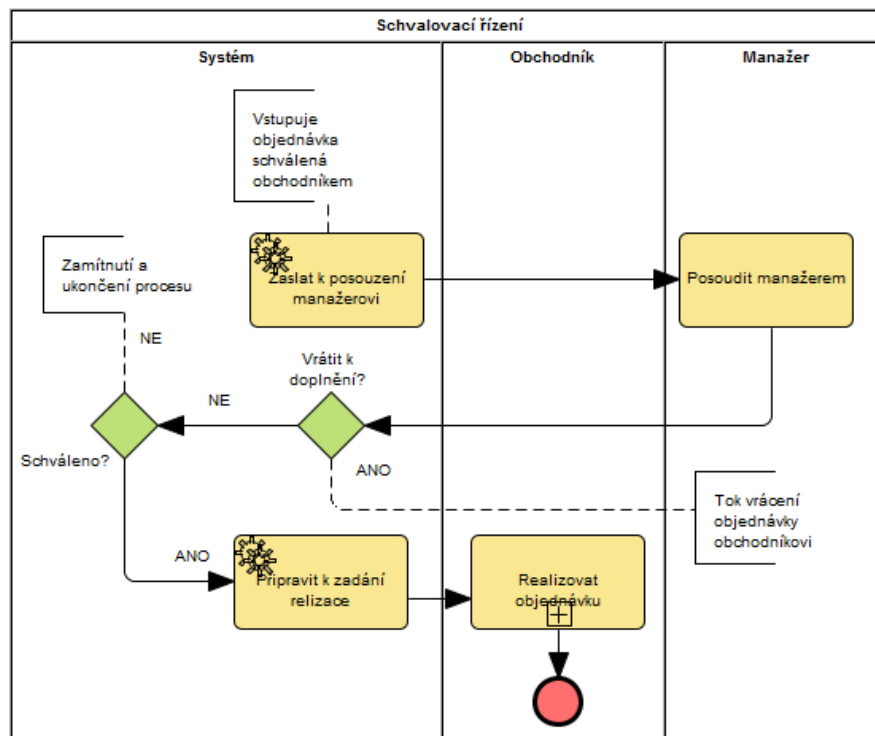


Obr. 11. Diagram aktivity Vedoucího oddělení

Klíčovou úlohu v procesu má *Obchodník*. V rámci řešeného procesu má nejvíce kompetencí i povinností. Jeho úkolem je řízení schvalovacího procesu a přijímání rozhodnutí v jeho průběhu (Obr. 12). Posuňme se do času, kdy objednávka s přiřazenými odbornými oblastmi je již kompletně posouzena všemi experty a vrací se zpět k vyhotovení stanoviska *Obchodníkem*. V této chvíli *Obchodník* nejprve rozhoduje, zda je požadavek po obsahové stránce úplný. Pokud vyhodnotí, že je nutné předložený materiál doplnit, potom objednávku vrací k doplnění. Určí, zda postačuje vyžádat doplnění odborných posudků experty,

vrátit zpět k doplnění. V případě takového rozhodnutí je objednávka vrácena obchodníkovi, který zváží další postup (viz Obr. 12).

Manažer má dále kompetenci schválit, či zamítnout objednávku. V případě schválení *Systém* objednávce nastaví příznak zaslání k realizaci a požadavek se vrací *Obchodníkovi* k zahájení procesu realizace.



Obr. 13. Diagram aktivity Manažera

4.1.3 Proces realizace objednávky

Prvním krokem v procesu realizace objednávky je přiřazení osoby odpovědné za nákup. *Obchodník* může realizaci zajistit sám nebo ji předat k vyřízení odborníkovi pro danou oblast, do které spadá předmět objednávky. Na diagramu (Příloha P IV) prochází tok procesu exkluzivní bránou s možností výběru jednoho směru pokračování. Přidělování je provedeno na konkrétního zaměstnance, který pověření musí potvrdit.

Jak jsem již popsal v analýze požadavků (str. 14), jednotlivé úkony realizace (poptávka, objednávka u dodavatele, nákup) jsou prováděny v interním logistickém systému nemocnice. Nejsou tak předmětem procesu schvalování objednávky, jsou však jeho logickým završením. V budovaném systému tedy bude existovat možnost shora popsaného přidělení řešitele, odpovědného za naplnění požadavku.

4.2 Funkční požadavky

Funkční požadavky definují požadovanou funkcionalitu aplikace jako reakci na podněty nebo situace. Celkový pohled na funkční požadavky je uveden v příloze (Příloha P V).

- R1. Systém povede evidenci uživatelů
- R2. Systém autentifikuje uživatele podle zadaného uživatelského jména a hesla
- R3. Systém eviduje emailové adresy uživatelů
- R4. Systém umožní provést obnovu zapomenutého hesla
- R5. Systém ukládá heslo v zabezpečeném tvaru pomocí hash algoritmu
- R6. Systém povede evidenci objednávek
- R7. Systém umožní přístup k vybrané objednávce na základě ověření oprávnění uživatele
- R8. Systém eviduje objednávky uživatele, umožní jejich zobrazení
- R9. Zobrazený seznam objednávek bude možné dále filtrovat do užšího výběru
- R10. Systém umožní vložit k objednávce stanovisko
- R11. K objednávce bude možné vyžádat konzultaci
- R12. Systém umožní vytvoření nové objednávky oprávněným uživatelem
- R13. K objednávce bude možné přiložit dokument ve formátu přílohy
- R14. Obsah objednávky budou tvořit položky objednávky. Systém umožní jejich vložení.
- R15. K objednávce bude možné přiřadit odbornou oblast
- R16. Systém poskytne informace o změně stavu objednávky
- R17. Systém bude generovat oznámení na určené emailové adresy
- R18. Systém umožní správu číselníků

4.3 Nefunkční požadavky

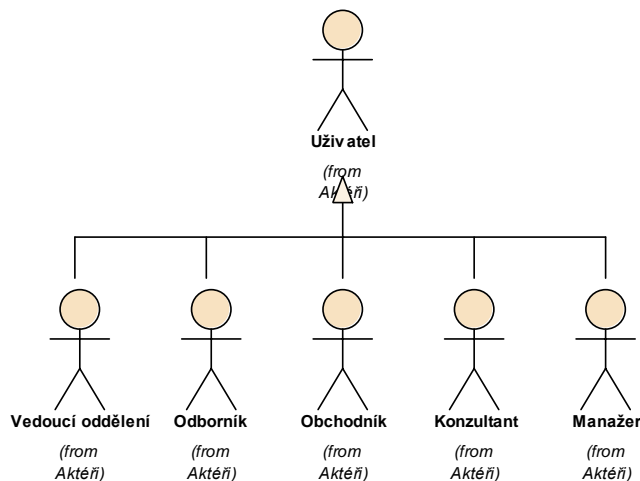
Nefunkční požadavky definují nároky a omezení systému.

- NR1. Systém bude vytvořen jako webová aplikace
- NR2. Systém poskytne vysokou míru zabezpečení
- NR3. Systém neumožní přístup uživateli bez přiděleného oprávnění

4.4 Diagramy případů užití a scénáře

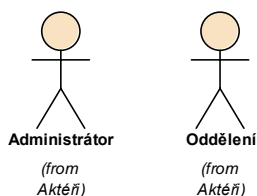
Mezi základní zdroje informací pro modelování případů užití patří model podnikového procesu a definice požadavků. Tyto informace máme již z předchozí části k dispozici a můžeme tak přistoupit k vytvoření diagramů případů užití.

V business process modelu jsou vyjmenováni základní aktéři systému. K nim v průběhu definování funkčních požadavků přibyl další aktér: *Konzultant*. Vybraní specializovaní aktéři dědí od obecného uživatele a můžou tak zakreslit vztah generalizace, viz obrázek (Obr. 14).



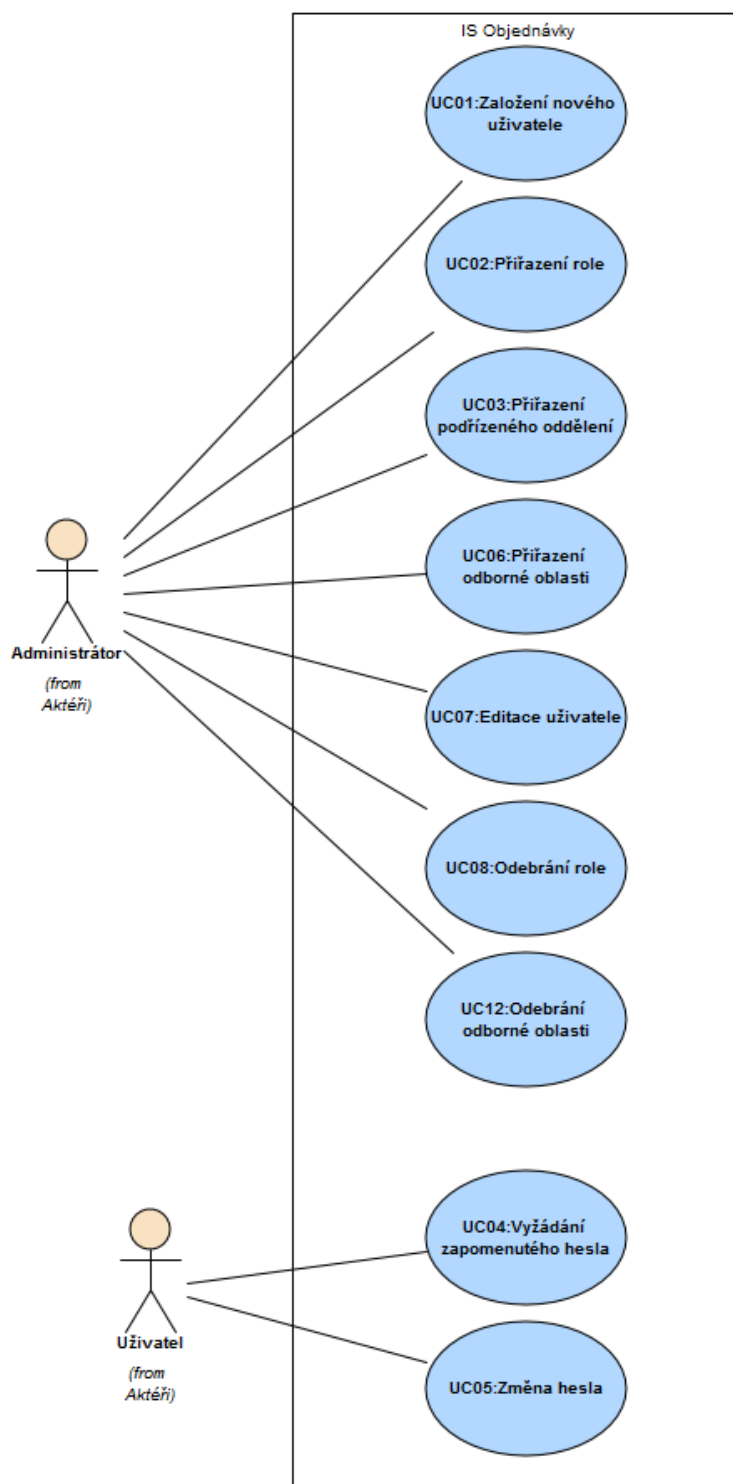
Obr. 14. Vztah generalizace aktérů systému

Dalšími aktéry jsou *Administrátor* a *Oddělení*, kteří mají v systému odlišné postavení od obecného Uživatele a jsou tak definováni samostatně (Obr. 15).

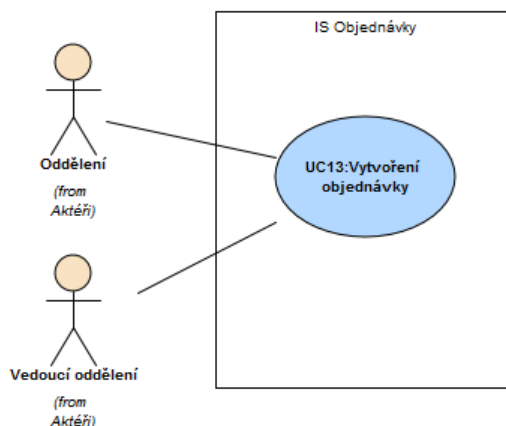


Obr. 15. Speciální aktéři

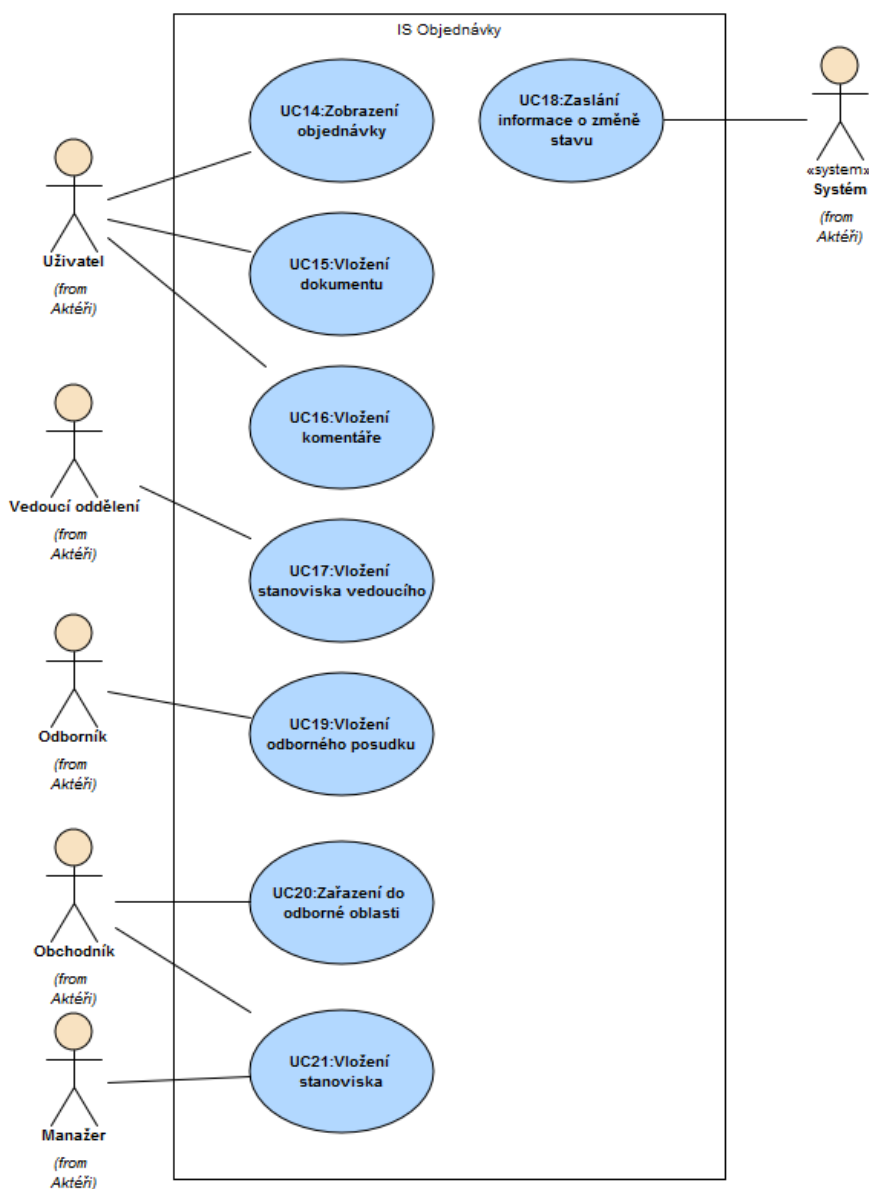
V následujících diagramech (Obr. 16 - Obr. 20) jsou zachyceny primární aktéři systému a k nim vázané případy užití. Jedná se o základní, a tedy hrubší pohled na modelovanou realitu tak, aby byla vyloučena nadbytečná dekompozice. Diagramy díky tomu zůstávají lépe srozumitelné. Přesnější určení vazeb mezi aktéry a jednotlivými scénáři je následně rozpracován v rozšířených diagramech případů užití, včetně relací <<include>>. Tento postup odpovídá doporučení dle [50].



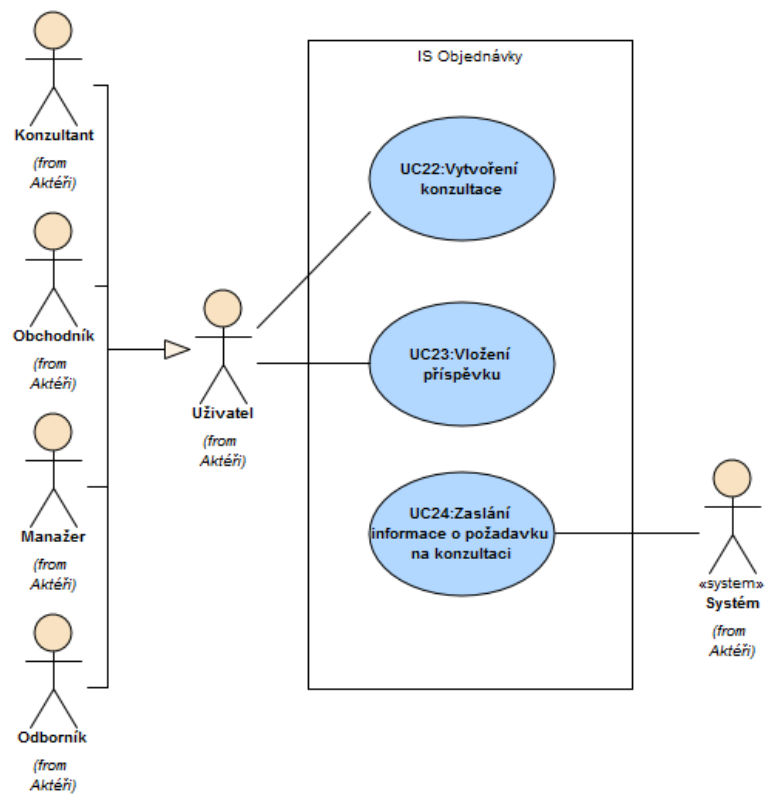
Obr. 16. Základní diagram případů užití administrace



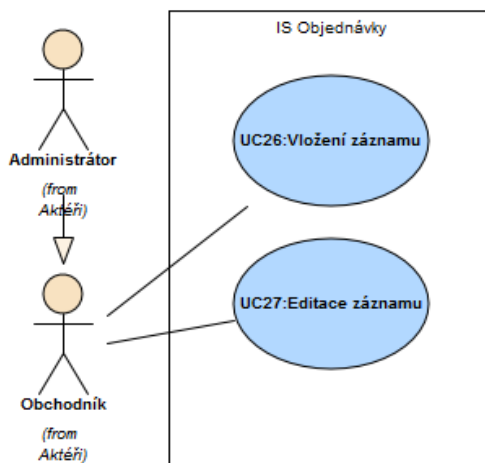
Obr. 17. Základní diagram případů užití založení objednávky



Obr. 18. Základní diagram případů užití obsluhy objednávky

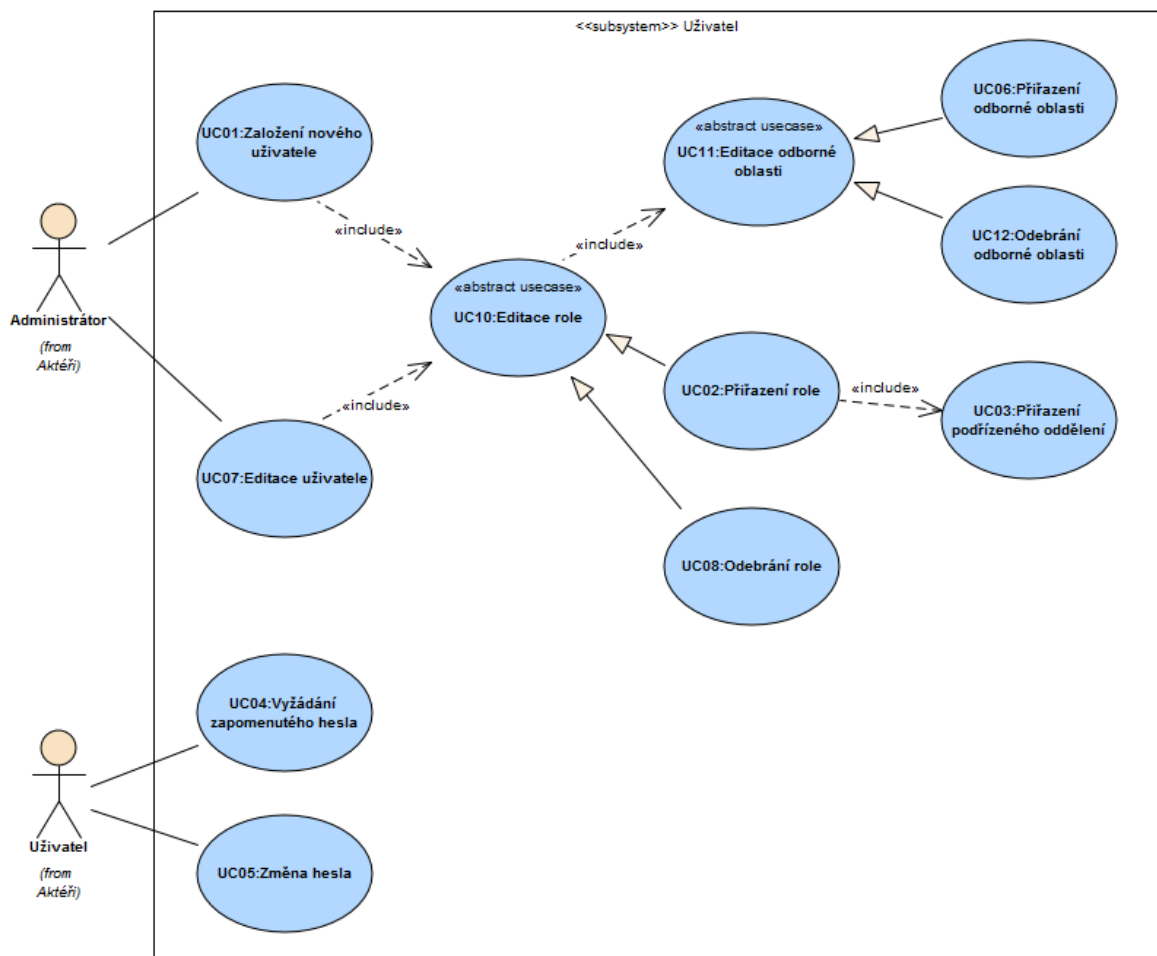


Obr. 19. Základní diagram případů užití oblasti konzultací



Obr. 20. Základní diagram případů užití správy číselníků

První rozšířený diagram případů užití je uveden na obrázku (Obr. 21). Aktéry scénářů uvedeného subsystému jsou *Administrátor* a *Uživatel*.



Obr. 21. Diagram případů užití Uživatel

4.4.1 UC01: Založení nového uživatele

Hlavní tok události:

1. Případ užití začíná, když Administrátor vybere volbu Založení nového uživatele
2. System zobrazí formulář pro vložení identifikačních údajů nového uživatele
3. Administrátor vyplní požadované údaje
4. Administrátor přiřadí uživateli role
 - a. INCLUDE UC02:Přiřazení role
5. IF role uživatele je „Vedoucí oddělení“
 - a. INCLUDE UC03:Přiřazení podřízeného oddělení
6. IF role uživatele je „Odborník“
 - a. INCLUDE UC06:Přiřazení odborné oblasti
7. Administrátor potvrdí vložené údaje
8. System vytvoří nového uživatele
9. Případ užití končí

4.4.2 UC02: Přiřazení role

Hlavní tok události:

1. Systém zobrazí seznam dostupných uživatelských rolí
2. FOR každou vybranou roli
 - a. Administrátor vybere roli a potvrdí výběr
 - b. Systém přiřadí uživateli vybranou roli

4.4.3 UC03: Přiřazení podřízeného oddělení

Hlavní tok události:

1. Systém zobrazí formulář pro přiřazení podřízeného oddělení
2. Administrátor vybere ze seznamu jedno oddělení a volbu potvrdí
3. Systém přiřadí zvolené NS uživateli

4.4.4 UC04: Vyžádání zapomenutého hesla

Hlavní tok události:

1. Příklad užití začíná, když Uživatel zobrazí stránku pro obnovu hesla
2. Systém zobrazí formulář pro zadání kontaktního emailu
3. Uživatel vyplní požadovaný údaj a volbu potvrdí
4. Systém ověří, zda v databázi existuje zvolený email a ztotožní uživatele
5. Systém vygeneruje autorizační token a společně s časem expirace jej uloží do databáze k záznamu uživatele
6. Systém na zvolený email zašle webový odkaz obsahující autorizační token
7. Uživatel zobrazí pomocí odkazu stránku pro změnu hesla
8. Systém zkontroluje autorizační token a porovná aktuální čas s časem expirace
9. Systém zobrazí formulář pro výměnu hesla
10. Uživatel vloží nové heslo a volbu potvrdí
11. Systém uloží změny
12. Příklad užití končí

Alternativní tok:

4a. Kontaktní email není nalezen

1. Systém zobrazí informaci o nenalezení kontaktního emailu
2. Příklad užití končí

8a. Autorizační token není správný nebo čas pro výměnu hesla expiroval

1. Systém zobrazí informaci, že není možné heslo vyměnit

2. Příklad užití končí

4.4.5 UC05: Změna hesla

Hlavní tok události:

1. Příklad užití začíná, když přihlášený uživatel zobrazí stránku pro změnu hesla
2. Systém zobrazí formulář pro změnu hesla
3. Uživatel vloží nové heslo a volbu potvrdí
4. Systém uloží změny
5. Příklad užití končí

4.4.6 UC06: Přiřazení odborné oblasti

Hlavní tok události:

1. Systém zobrazí formulář pro přiřazení odborných oblastí
2. FOR každou vybranou odbornou oblast
 - a. Administrátor vybere odbornou oblast a potvrdí výběr
 - b. Systém přiřadí uživateli vybranou odbornou oblast

4.4.7 UC07: Editace uživatele

Hlavní tok události:

1. Příklad užití začíná, když Administrátor vybere volbu Editovat uživatele
2. Systém zobrazí stránku s detaily vybraného uživatele
3. Administrátor upraví požadované údaje
4. Administrátor upraví uživateli role
 - a. INCLUDE UC02:Přiřazení role
 - b. INCLUDE UC08:Odebrání role
5. IF role uživatele je „Vedoucí oddělení“
 - a. INCLUDE UC03:Přiřazení podřízeného oddělení
6. IF role uživatele je „Odborník“
 - a. INCLUDE UC06:Přiřazení odborné oblasti
 - b. INCLUDE UC12:Odebrání odborné oblasti
7. Administrátor potvrdí vložené údaje
8. Systém uloží provedené změny
9. Příklad užití končí

4.4.8 UC08: Odebrání role

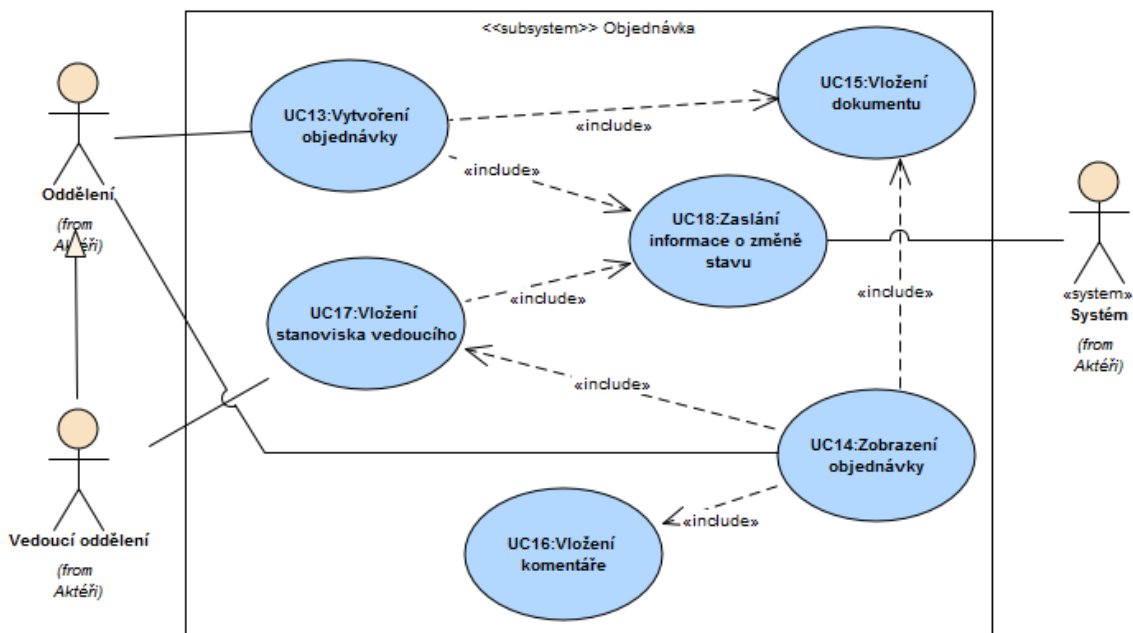
Hlavní tok události:

1. Systém zobrazí seznam přidělených uživatelských rolí
2. FOR každou vybranou roli
 - a. Administrátor vybere roli a potvrdí výběr
 - b. Systém odebere uživateli vybranou roli

4.4.9 UC12: Odebrání odborné oblasti

Hlavní tok události:

1. Systém zobrazí formulář pro odebrání odborných oblastí
2. FOR každou vybranou odbornou oblast
 - a. Administrátor vybere jednu z přidělených odborných oblastí a potvrdí výběr
 - b. Systém odebere uživateli vybranou odbornou oblast



Obr. 22. Diagram případů užití Objednávka

4.4.10 UC13: Vytvoření objednávky

Hlavní tok události:

1. Případ užití začíná, když uživatel *Oddělení* nebo *Vedoucí oddělení* vybere volbu vytvoření objednávky
2. Systém zobrazí formulář pro vložení údajů k objednávce
3. IF uživatel vybere dokument, který se má připojit k objednávce
 - a. INCLUDE UC15: Vložení dokumentu
4. Uživatel potvrdí zapsané údaje

5. Systém uloží objednávku
6. INCLUDE UC18:Zaslání informace o změně stavu objednávky
7. Příklad užití končí

4.4.11 UC14: Zobrazení objednávky

Hlavní tok události:

1. Příklad užití začíná, když uživatel vybere volbu zobrazení objednávky
2. Systém zobrazí vybranou objednávku
3. IF uživatel vybere dokument, který se má připojit k objednávce
 - a. INCLUDE UC15:Vložení dokumentu
4. IF uživatel zvolí možnost vložit komentář k objednávce
 - a. INCLUDE UC16:Vložení komentáře
5. IF role uživatele je „Vedoucí oddělení“
 - a. INCLUDE UC17:Vložení stanoviska vedoucího oddělení
 - b. INCLUDE UC18:Zaslání informace o změně stavu objednávky
6. Příklad užití končí

4.4.12 UC15: Vložení dokumentu

Hlavní tok události:

1. FOR každý vybraný dokument
 - a. Systém načte data dokumentu
 - b. Systém uloží dokument do databáze

4.4.13 UC16: Vložení komentáře

Hlavní tok události:

1. Uživatel vloží do formuláře svůj komentář k objednávce
2. Systém vložené údaje uloží

4.4.14 UC17: Vložení stanoviska vedoucího oddělení

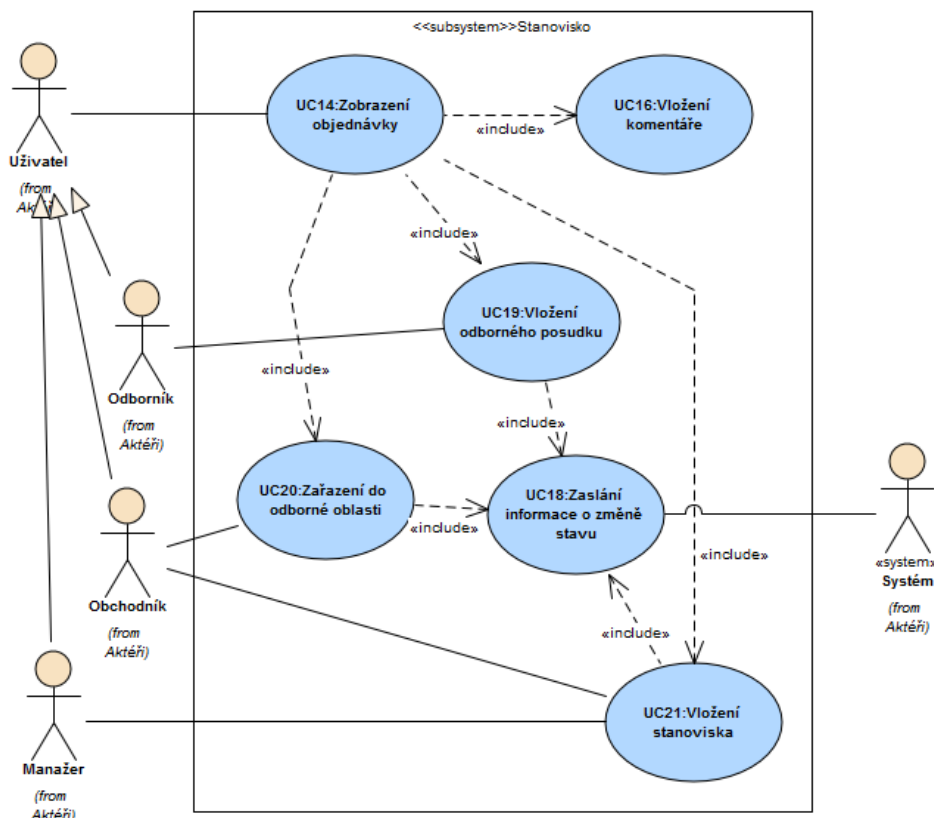
Hlavní tok události:

1. IF je vyžádáno stanovisko vedoucího k objednávce
 - a. Systém zobrazí formulář pro vložení stanoviska k objednávce
 - b. Vedoucí oddělení vloží své stanovisko k objednávce
2. INCLUDE UC18:Zaslání informace o změně
3. Příklad užití končí

4.4.15 UC18: Zaslání informace o změně stavu

Hlavní tok události:

1. Systém podle typu změny stavu
 - a. Z databáze vybere email adresy příjemců zprávy
 - b. Zvolí šablonu zprávy
2. Systém vygeneruje zprávu a rozešle adresátům
3. Příklad užití končí



Obr. 23. Diagram případů užití Stanovisko

4.4.16 UC19: Vložení odborného posudku

Hlavní tok události:

1. IF uživatel je „Odborník“
 - a. Systém zobrazí formulář pro vložení odborného posudku
 - b. Uživatel zadá data a potvrdí vložení
 - c. INCLUDE UC18: Zaslání informace o změně stavu
2. Příklad užití končí

4.4.17 UC20: Zařazení do odborné oblasti

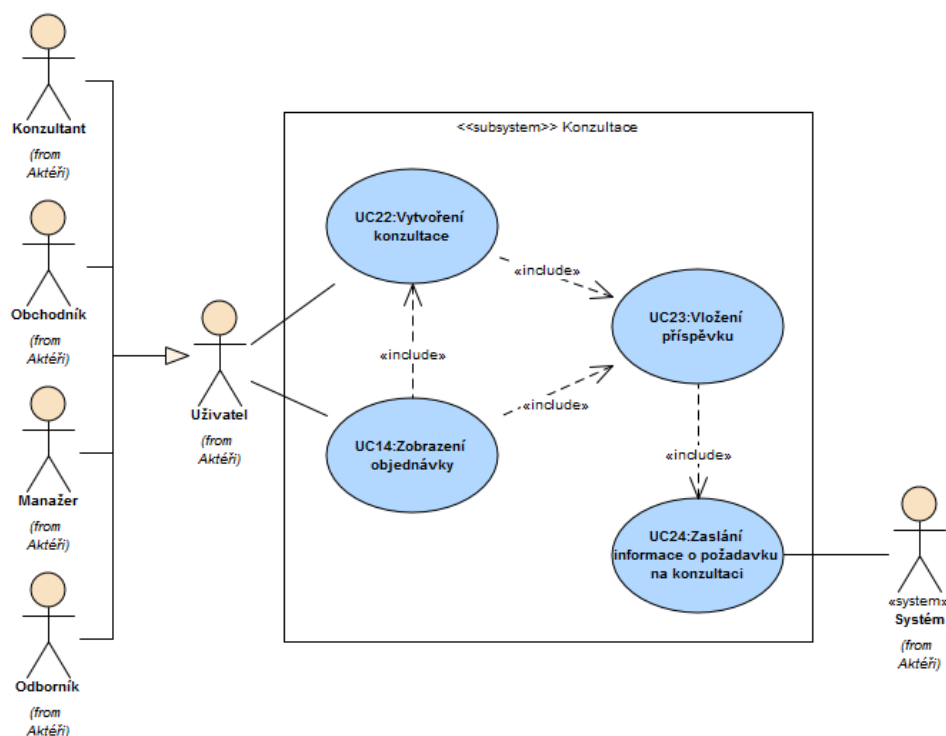
Hlavní tok události:

1. IF uživatel je „Obchodník“
 - a. Systém zobrazí formulář pro přiřazení odborné oblasti
 - b. Uživatel vybere požadované odborné oblasti a údaje uloží
 - c. INCLUDE UC18:Zaslání informace o změně stavu
2. Příklad užití končí

4.4.18 UC21: Vložení stanoviska

Hlavní tok události:

3. IF uživatel je „Obchodník“ nebo „Manažer“
 - a. Systém zobrazí formulář pro vložení stanoviska k objednávce
 - b. Uživatel vyplní údaje a změny uloží
 - c. INCLUDE UC18:Zaslání informace o změně stavu
4. Příklad užití končí



Obr. 24. Diagram případů užití Konzultace

4.4.19 UC22: Vytvoření konzultace

Hlavní tok události:

1. Uživatel zobrazí objednávku, kterou požaduje konzultovat
2. Systém zobrazí formulář pro výběr adresáta a vložení textu problematiky
3. Uživatel vyplní údaje a potvrdí uložení
4. Systém založí hlavičku konzultace a první příspěvek
 - a. INCLUDE UC23:Vložení příspěvku
5. INCLUDE UC24:Zaslání informace o požadavku na konzultaci
6. Příklad užití končí

4.4.20 UC23: Vložení příspěvku

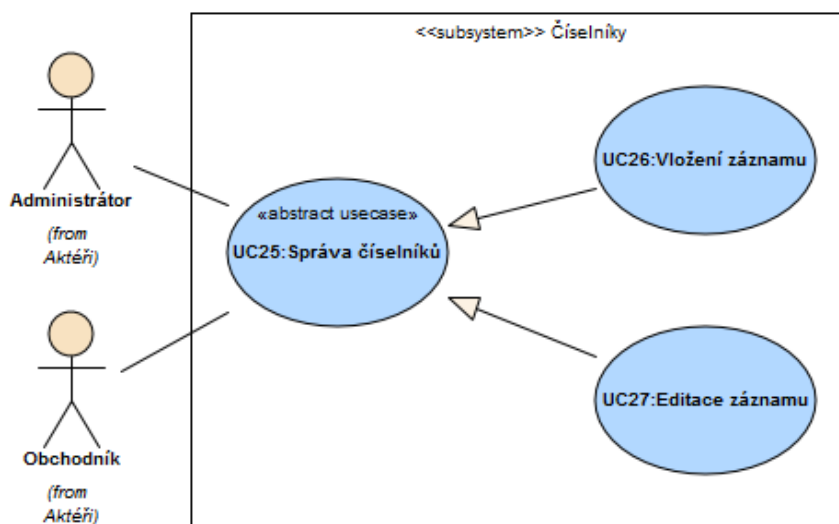
Hlavní tok události:

1. Uživatel zobrazí objednávku, ke které je vyžádána jeho konzultace
2. Systém zobrazí formulář pro vložení příspěvku
3. Uživatel vyplní údaje a potvrdí uložení
4. INCLUDE UC24:Zaslání informace o požadavku na konzultaci
5. Příklad užití končí

4.4.21 UC24: Zaslání informace o požadavku na konzultaci

Hlavní tok události:

1. Systém z databáze vybere emailovou adresu příjemce zprávy
2. Systém vygeneruje zprávu o vložení nového příspěvku do konzultace a rozešle adresátům
3. Příklad užití končí



Obr. 25. Diagram případů užití Číselníky

4.4.22 UC25: Správa číselníků

Hlavní tok události:

1. Uživatel vybere z menu požadovaný číselník
2. Uživatel zvolí typ úkonu, který požaduje provádět nad číselníkem
3. IF vybraný úkon je vložení záznamu
 - a. INCLUDE UC26:Vložení záznamu
4. IF vybraný úkon je editace záznamu
 - a. Uživatel zvolí konkrétní záznam
 - b. INCLUDE UC27>Editace záznamu
5. Příklad užít končí

4.4.23 UC26: Vložení záznamu

Hlavní tok události:

1. Systém zobrazí formulář pro vložení nového záznamu
2. Uživatel vyplní údaje a potvrdí uložení
3. Příklad užití končí

4.4.24 UC27: Editace záznamu

Hlavní tok události:

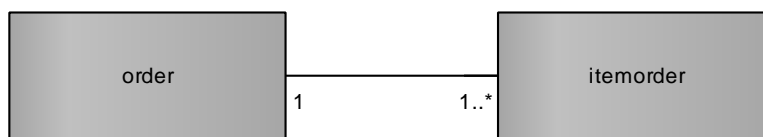
1. Systém zobrazí formulář pro editaci záznamu
2. Uživatel upraví požadované údaje a potvrdí uložení
3. Příklad užití končí

5 NÁVRH ZÁKLADNÍCH ENTIT

V předchozí části byly definovány požadavky, které musí informační systém splňovat a modelovány diagramy případů užití. Na základě těchto znalostí vytvořím ER model¹². Nejprve určím základní entity a definuju vazby mezi těmito entitami. V dalším kroku, což bude návrh rozšířeného entitně relačního modelu, pak zformuluju jednotlivé atributy a klíče entit.

5.1 Objednávka a její položky

Primárním objektem zájmu bude objednávka, jež se skládá z jednotlivých položek a je možné ji popsat dvěma entitami *order* a *itemorder*. Entita *order* obsahuje identifikační (hlavičkové) údaje objednávky a entita *itemorder* představuje předmět objednávky. Tedy konkrétní požadované zboží či služby. Vztah mezi entitami *order* - *itemorder* bude $(1, n) : (1, 1)$.

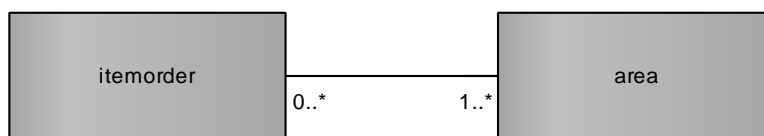


Obr. 26. Entita objednávky a položky objednávky

5.2 Odborná oblast

Každá položka objednávky bude v průběhu schvalovacího procesu odborně posuzována, a to z pohledu možných způsobů vykrytí požadavku, resp. z pohledu potřebnosti a účelnosti požadovaného. Pro určení v jaké odborné oblasti mají být položky objednávky posouzeny, bude každá z nich jednotlivě zařazena do jedné či více odborných oblastí. Odborné oblasti budou popsány entitou *area*. Vztah mezi entitami *itemorder* a *area* bude $(1, n) : (0, n)$.

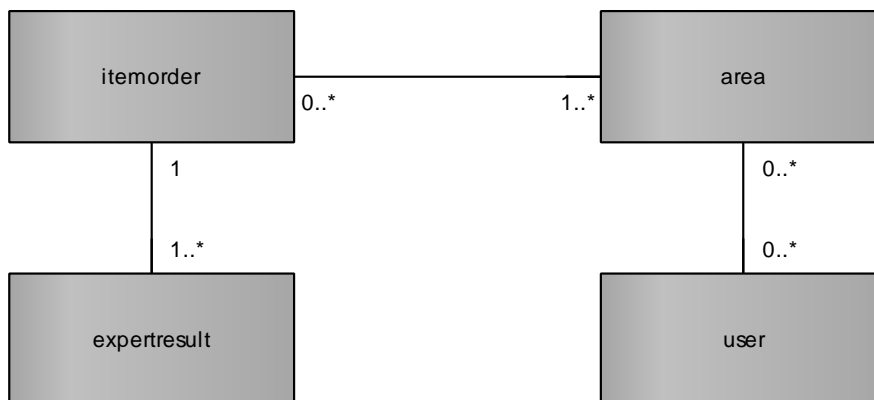
¹² ER model – entitně-relační model databáze definuje její konceptuální schéma. Je založen na dvou typech objektů – entity a vztahy mezi nimi.



Obr. 27. Entita pro zařazení položky do odborné oblasti

5.3 Odborné posudky objednávky

Odborné posudky budou v systému reprezentovány entitou *expertresult*. Jak již bylo uvedeno výše, každá položka objednávky je povinně zařazena nejméně do jedné odborné oblasti, ve které má být posuzována. V informačním systému budou zároveň definovány uživatelé (experti), kteří na základě přidělené odborné oblasti budou mít právo vystavit položce objednávky odborný posudek. Entita *expertresult* bude ve vztahu (1, 1) : (1, n) s entitou *itemorder*. Entita *user*, která bude podrobněji popsána v samostatné kapitole, bude mít s entitou *area* vztah (0, n) : (0, n).

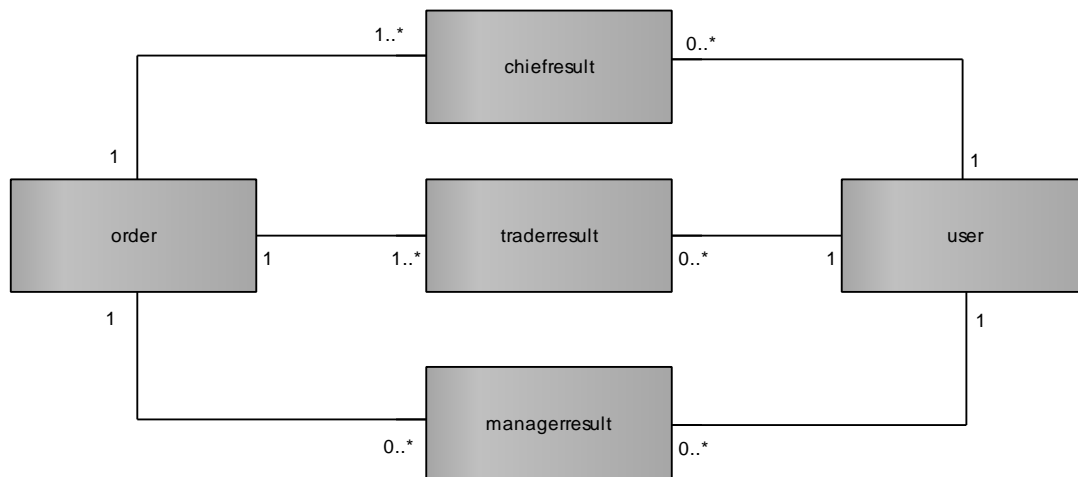


Obr. 28. Entita pro záznam odborného posudku

5.4 Stanoviska k objednávce

Proces schvalování objednávky má sekvenční průběh zpracování. Postupuje se po jednotlivých úsecích, kdy jednotliví aktéři vložím svého stanoviska do systému posunou schvalovací proces do další fáze. Tyto klíčové aktéry představuje vedoucí žádajícího oddělení, pracovník obchodního oddělení, odborník a manažer. Mimo odborníka, jehož posudek se vztahuje k jednotlivým položkám objednávky, se ostatní aktéři vyjadřují k objednávce jako

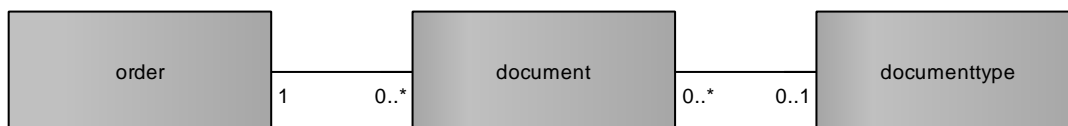
celku. Stanoviska těchto aktérů bude popisovat entita *chiefresult*, *traderresult*, *managerresult*. Entita *order* bude s entitou *chiefresult* a *traderresult* ve vztahu (1, n) : (1, 1). S entitou *managerresult* bude ve vztahu (0, n) : (1, 1). Tento vztah je odlišný tím, že ne ve všech případech bude objednávka posouzena i manažerem, může být zamítnuta již na úrovni pracovníka obchodního oddělení. Entita *user* bude ve vztahu (0, n) : (1, 1) s entitami *chiefresult*, *traderresult* a *managerresult*.



Obr. 29. Entita pro záznam stanoviska k objednávce

5.5 Dokumenty

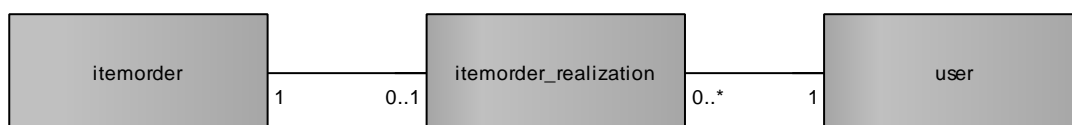
Dokumenty evidované v systému bude představovat entita *document*. Dokumenty budou součástí objednávky jako její příloha. Mohou je reprezentovat obchodní nabídky, manuály nebo i faktury, za již zakoupené zboží či služby. Volitelně je možné dokumenty rozlišit typem, který lze dokumentům jednotlivě přiřazovat. Typ dokumentu bude popsán entitou *documenttype*. Vztah mezi entitami *order* a *document* bude (0, n) : (1, 1). Entita *document* bude s entitou *documenttype* ve vztahu (0, 1) : (0, n).



Obr. 30. Entita pro dokument a typ dokumentu

5.6 Záznam o realizaci objednávky

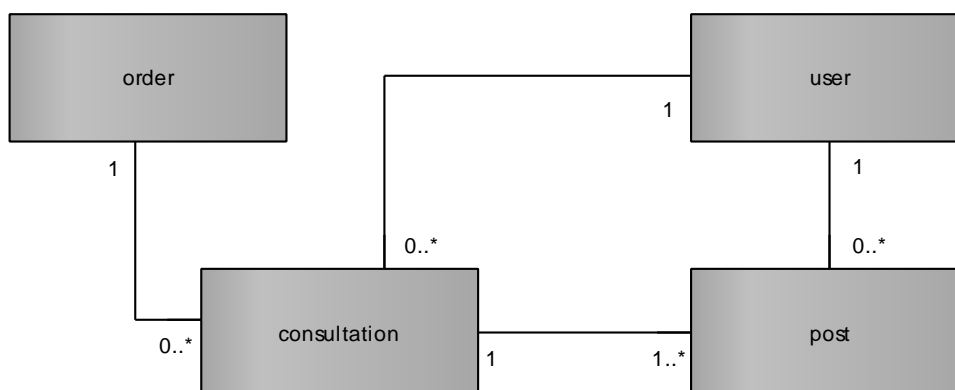
Objednávka může být vyřízena pro všechny své položky, částečně, případně může být zamítnuta. S ohledem k možnosti částečného plnění bude záznam o realizaci objednávky popisovat vyřízení každé jednotlivé položky objednávky. Záznam realizace bude představovat entita *itemorder_realization*. Realizací schválené objednávky bude pověřen konkrétní uživatel. Vztah *itemorder_realization* a *itemorder* bude typu $(1, 1) : (0, 1)$. Vztah mezi entitou *itemorder_realization* a *user* bude $(1, 1) : (0, n)$.



Obr. 31. Entita pro záznam realizace objednávky

5.7 Konzultace k objednávce

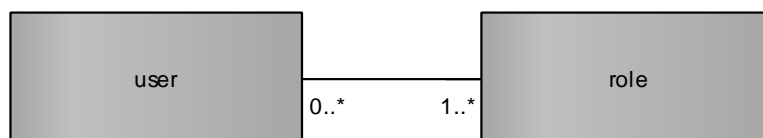
Konzultace vyjadřuje podpůrnou činnost v procesu schvalování objednávky. V libovolné fázi zpracování je možné vyžádat od konkrétního aktéra vyjádření k předmětu objednávky. Konzultace budou tvořeny entitami *consultation* a *post*. Jejich vzájemný vztah bude $(1, n) : (1, 1)$. Entita *consultation* bude dále ve vztahu $(1, 1) : (0, n)$ s entitami *order* a *user*. Entita *post* bude obsahovat příspěvky v dané konzultaci a bude ve vztahu $(1, 1) : (0, n)$ s entitou *user*.



Obr. 32. Entita konzultace k objednávce

5.8 Uživatel a uživatelské role

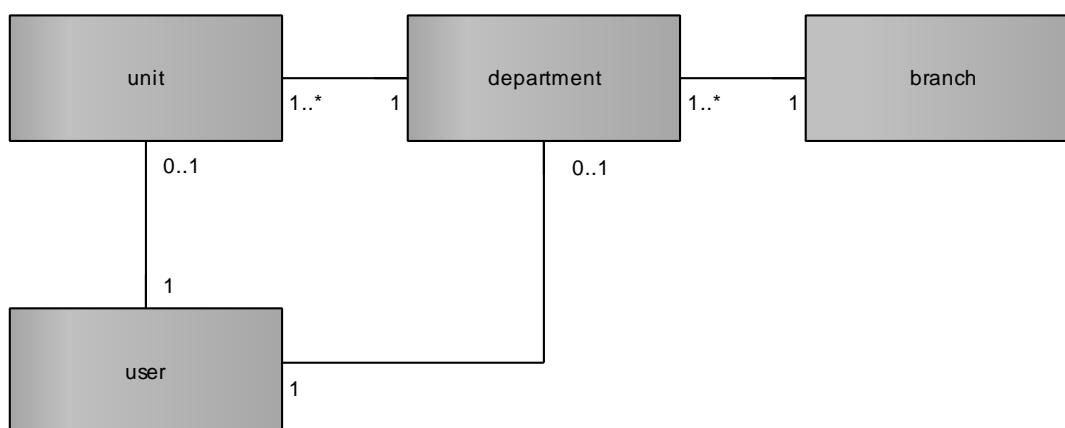
Uživatele systému lze popsat entitou *user*, která je s entitou *role* ve vztahu $(1, n) : (0, n)$. Entita *role* představuje uživatelské oprávnění. Záznam o uživateli bude vyjadřovat konkrétní osobu, fyzického aktéra systému, nebo se bude jednat o definici pracoviště – nákladového střediska.



Obr. 33. Entita uživatele a jeho role

5.9 Organizační členění

Nemocnice se člení na obory, oddělení, nákladová střediska. V systému je budou představovat entity *branch*, *department*, *unit*. Jak už bylo uvedeno výše, entita *user* vyjadřuje mimo jiné i konkrétní pracoviště nemocnice. Vztah mezi entitou *unit* a *user* bude $(1, 1) : (0, 1)$. Entita *department* popisuje oddělení nemocnice a s entitou *unit* bude ve vztahu $(1, n) : (1, 1)$, s entitou *branch*, která reprezentuje obor nemocnice, bude ve vztahu $(1, 1) : (1, n)$. Každé oddělení má definovaného svého vedoucího. Tento vztah je zachycen vazbou $(0, 1) : (1, 1)$ mezi entitami *user* a *department*.



Obr. 34. Entity vyjadřující organizační členění nemocnice

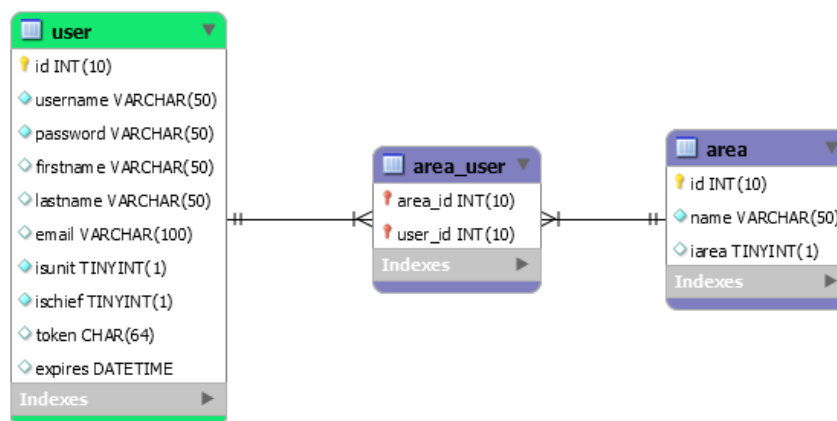
6 ROZŠÍŘENÝ ENTITNĚ RELAČNÍ MODEL DATABÁZE

Rozšířený entitně relační model (EER model) má již relace mezi entitami fyzicky realizovány pomocí cizích klíčů a vazebních tabulek. Při jeho tvorbě vycházím z poznatků, které byly zformulovány v předchozí části tvorby ER modelu. Schémata jednotlivých částí jsou uvedena na vložených obrázcích (Obr. 35 - Obr. 41). Celkové schéma se nachází v příloze (Příloha P II). Primární klíč většiny entit tvoří atribut *id*. Jedná se o bezznaménkové (unsigned) celé číslo typu INT, s nastavenou vlastností *Auto Increment*. Tato vlastnost zajišťuje automatické číslování primárního klíče. Cizí klíče jsou tvořeny zpravidla atributem pojmenovaným *<entita>_id*. Jak jsem již popsal v teoretické části, tato konvence vychází z implementace rozhraní `Nette\Database\IConventions`, které analyzuje cizí klíče a umožňuje jednoduše spravovat vztahy mezi tabulkami.

Relace (1, 1) : (0, n) jsou realizovány tak, že zdrojová tabulka obsahuje přímo cizí klíč cílové tabulky. Stejným způsobem je postupováno pro relaci (0, 1) : (0, n).

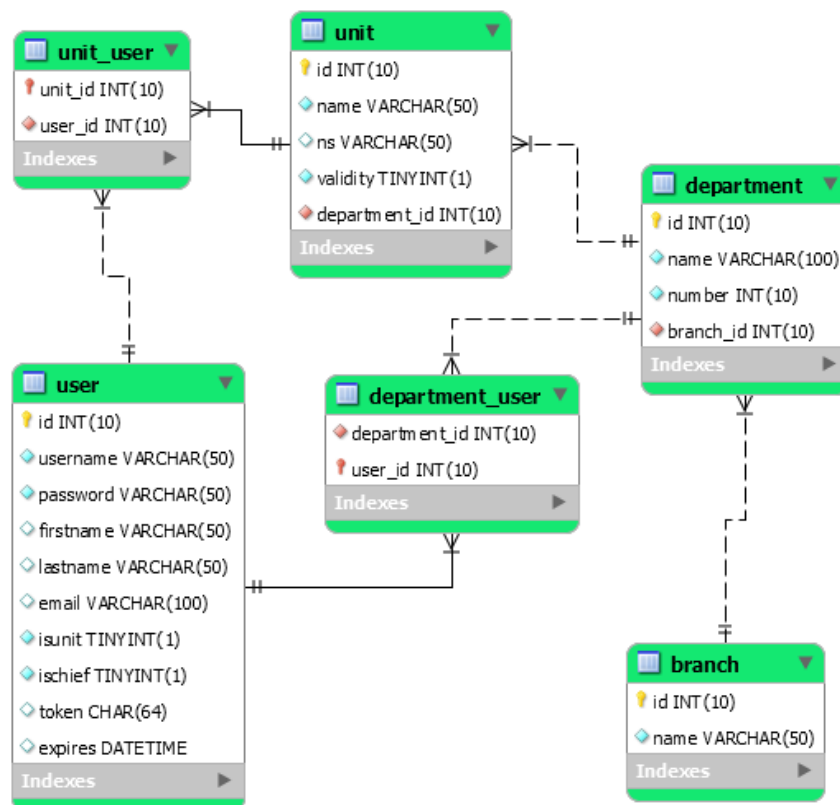
Relace (0, n) : (0, n) mezi entitami jsou realizovány za pomoci vazební tabulky, která obsahuje dva atributy. Atributy jsou tvořeny cizím klíčem do zdrojové tabulky a cizím klíčem do cílové tabulky. Ve většině případů společně tvoří složený primární klíč.

Příkladem tohoto spojení můžou být entity *user* a *area*. Entita *area* obsahuje záznamy o existujících odborných oblastech. Jednotlivé oblasti mohou být přiřazeny uživateli, který tak získá právo vystavit v dané oblasti odborný posudek k objednávce (položka objednávky má obdobnou vazbu na entitu *area*). Entita *user* obsahuje záznamy uživatelů. Vazební tabulka *user_area* má tedy jako první atribut (*area_id*) cizí klíč do tabulky *area*. Druhý atribut (*user_id*) je tvořen cizím klíčem do tabulky *user* (Obr. 35).



Obr. 35. EER schéma uživatelů a odborných oblastí

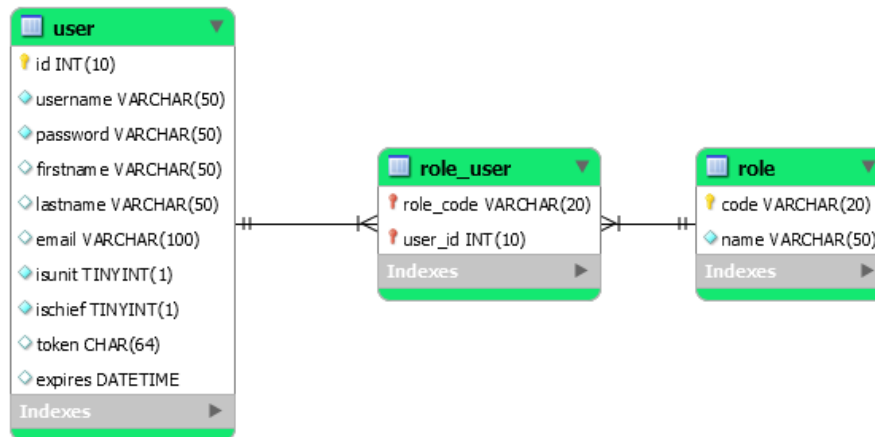
Organizační členění nemocnice je vrcholově rozděleno na zdravotnickou a nezdravotnickou část. Podle tohoto dělení se nejvyšší uzel hierarchie nazývá Obor, resp. Odbor. Další členění je pak již společné: Obor/Odbor → Oddělení → Nákladové středisko. Tato struktura je v databázi vyjádřena tabulkami *branch*, *department*, *unit*. Informační systém rozeznává dva základní typy uživatelů. Jednak jsou to fyzické osoby a v druhém případě organizační jednotky. Záznamy o uživatelích jsou uloženy v tabulce *user*, která je přes vazební tabulku *unit_user* propojena do tabulky *unit*. Každá jednotka (záznam z tabulky *unit*) má v tabulce *user* právě jeden záznam. Relace spojení je typu (1, 1) : (0, 1). Ve vazební tabulce je atribut *unit_id* primárním klíčem. Tím je zabezpečeno, že záznam o jednotce bude propojen vždy jen na jeden záznam o uživateli.



Obr. 36. EER schéma organizačního členění

Tabulka *user* je dále propojena na tabulku *department* přes vazební tabulku *department_user*. Toto spojení se uplatňuje v případě, že záznam fyzické osoby v roli vedoucího oddělení z tabulky *user* potřebujeme propojit se záznamem jemu podřízeného oddělení (Obr. 36).

Na schématu uživatelů a uživatelských rolí (Obr. 37) je zachycena vazba mezi entitami *user* a *role*. Každému uživateli je možné přidělit jednu či více rolí. Relace je tvořena vazební tabulkou *role_user*.



Obr. 37. EER schéma uživatelů a uživatelských rolí

Základní entity uvedené ve schématu (Obr. 38) tvoří jádro celé aplikace. Tabulka *order* je tvořena atributy, kterými jsou popsány hlavičkové údaje objednávky. Mezi tyto údaje patří název objednávky (*name*), který stručně vyjadřuje předmět objednávky, dále jsou to předpokládaná cena (*price*), požadovaný termín dodání (*deadline*), typ objednávky (*ordertype_id*), požadující (*user_id*), odůvodnění (*justify_id*) a forma realizace (*realization_id*). Pomocí jmenovaných cizích klíčů je tabulka *order* v relaci (1, 1) : (1, n) s tabulkami *ordertype*, *justify*, *realization*. Tyto tabulky jsou naplněny předdefinovanými záznamy, kterými jsou v aplikačním rozhraní naplněny příslušné rozbalovací seznamy.

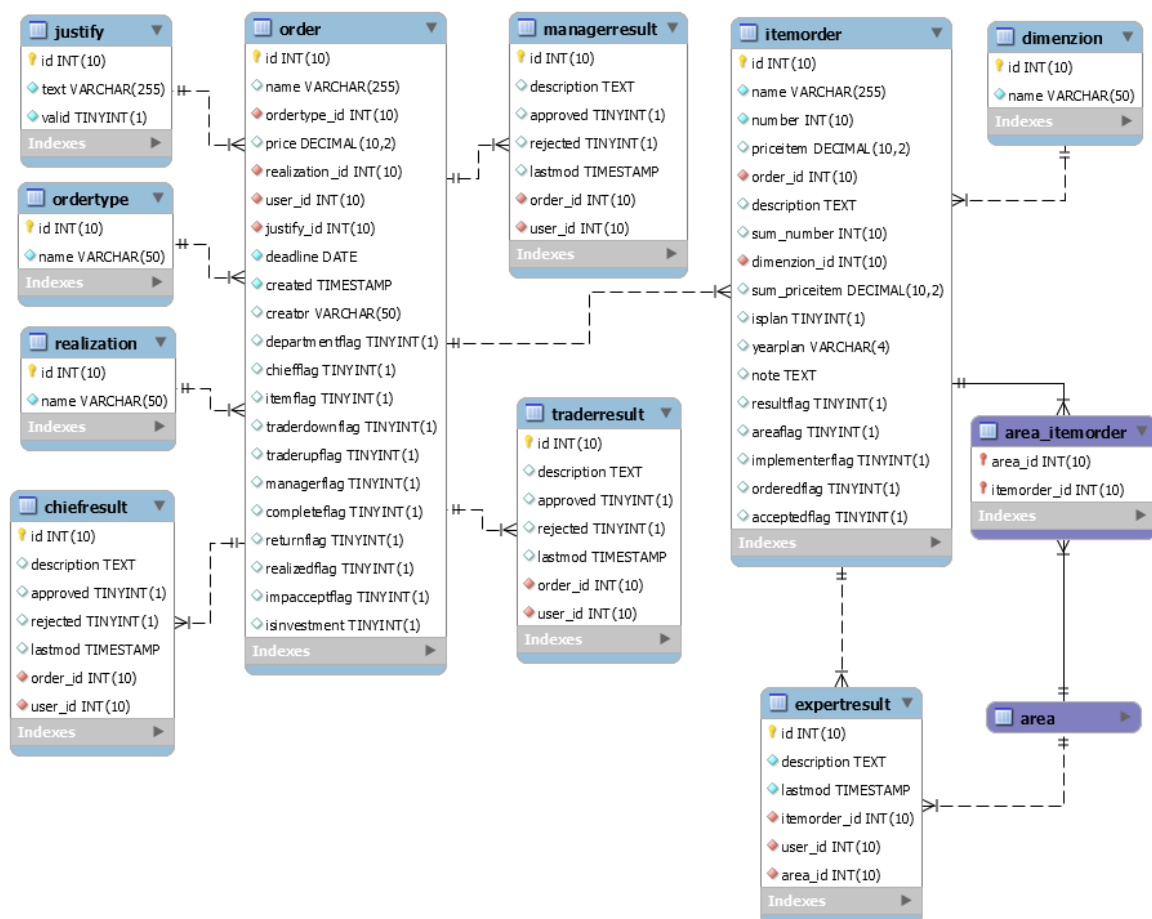
Tabulka *order* dále obsahuje deset atributů *<prefix>flag*. Tyto atributy jsou typu TINYINT a nabývají zpravidla hodnot NULL, 1 nebo 2. Jak bylo popsáno v části modelování business procesu, každá objednávka povinně prochází definovanými kroky schvalovacího procesu. Postup zpracování je zaznamenáván pomocí uvedených atributů. Speciálním atributem je atribut *returnflag*, který slouží k zachycení situace, kdy je objednávka vrácena k dopracování.

Tabulka *itemorder* je spojena vazbou N:1 s tabulkou *order*. V tabulce *itemorder* jsou záznamy o jednotlivých položkách objednávky. Konkretizuje se zde požadovaný předmět (*name*), množství (*number*) a cena (*priceitem*). Pomocí cizího klíče do tabulky *dimenzion* je uvedena jednotka množství. Dále se zde nachází atribut pro vložení popisu požadavku (*description*) a pomocné atributy, které slouží uživateli v roli Obchodník k uložení doplňkových informací k položce objednávky – *sum_number*, *sum_priceitem*, *isplan*, *yearplan* a

note. K zaznamenání průběhu zpracování položky objednávky jsou použity atributy $\langle prefix \rangle flag$, obdobně jako u tabulky *order*.

K uložení stanovisek k objednávce slouží tabulky *chiefresult*, *traderresult*, *managerresult*, *expertresult*. První tři vyjmenované tabulky jsou ve vazbě N:1 s tabulkou *order*. Pověření uživatelé s rolí Vedoucí oddělení, Obchodník, Manažer se povinně vyjadřují k objednávce jako k celku. Naproti tomu uživatel s rolí Expert vydává odborné posudky k jednotlivým položkám objednávky. Tabulka *expertresult* je proto ve vazbě N:1 s tabulkou *itemorder*.

K přiřazení odborných oblastí k položce objednávky je použita vazební tabulka *area_itemorder*. Každá z jednotlivých položek objednávky může být zařazena do libovolného počtu odborných oblastí.



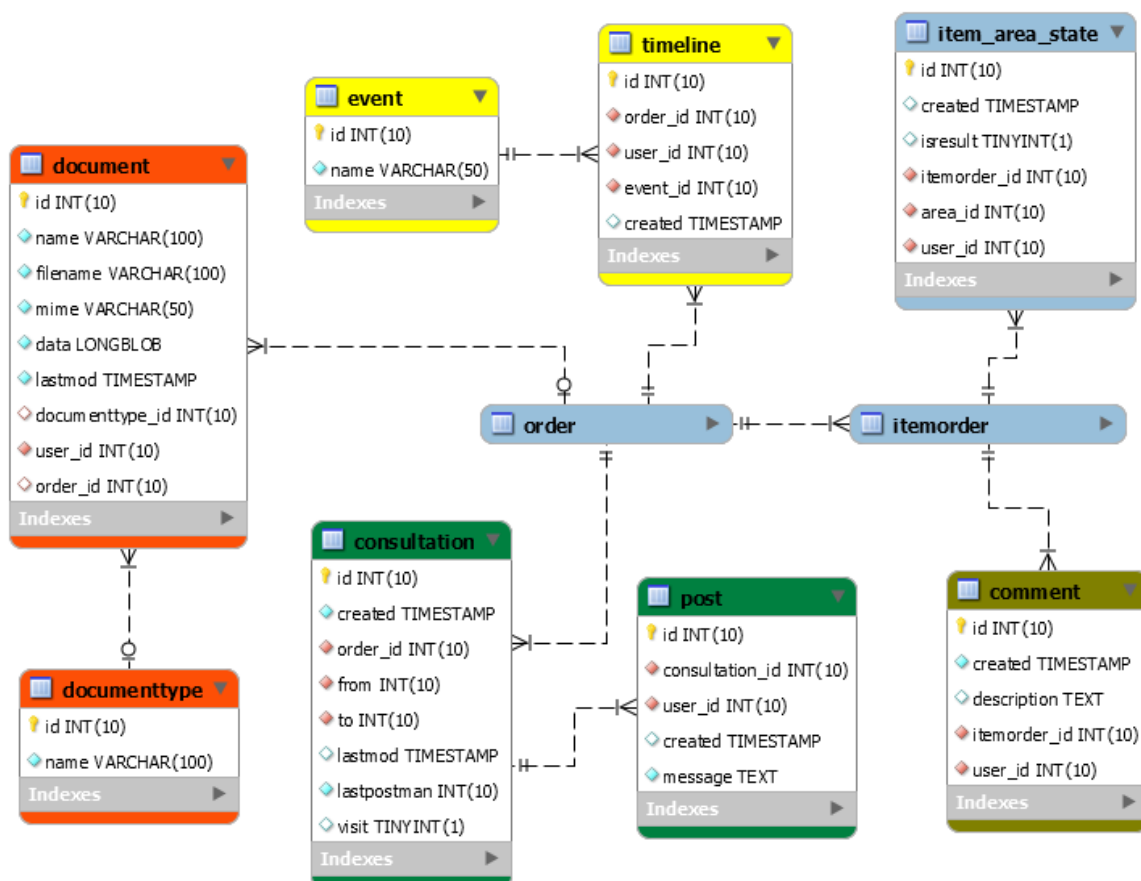
Obr. 38. EER schéma základních entit

Na obrázku (Obr. 39) je znázorněna vazba základních entit na entity, které obsahují přidružené údaje k objednávce. Entita *timeline* obsahuje záznamy o časovém průběhu schvalovacího řízení. Atribut *event_id* je cizím klíčem do tabulky *event*, která eviduje předvolené záznamy událostí, které nastávají v procesu zpracování objednávky.

K objednavce lze vyžádat konzultaci. Záznam o konzultaci je uložen do tabulky *consultation*. Jednotlivé příspěvky konzultace se vkládají do tabulky *post*. Ke každé položce objednávky je možné vkládat také komentáře. Pro jejich uložení se využívá tabulka *comment*.

Do tabulky *item_area_state* se ukládá záznam o přiřazení položky objednávky do odborné oblasti. Atribut *isresult* slouží k uvedení příznaku, zda položka byla pro příslušnou odbornou oblast posouzena.

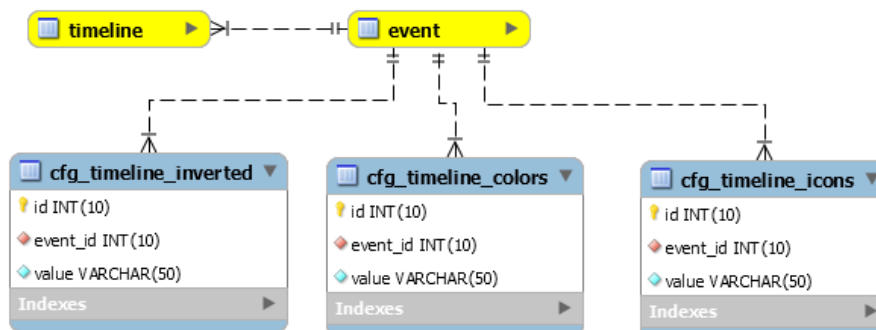
Evidované dokumenty jsou ukládány přímo do databáze. K tomu slouží tabulka *document*. Informace o dokumentu se ukládají pomocí tří atributů. Atribut *filename* obsahuje název souboru, atribut *data* jsou zdrojová data souboru a atribut *mime* udává typ dokumentu. MIME¹³ se používá pro rozlišení typu souboru. Např. MIME typ *image/png* odpovídá obrázku ve formátu PNG.



Obr. 39. EER schéma přidružených záznamů k objednávce

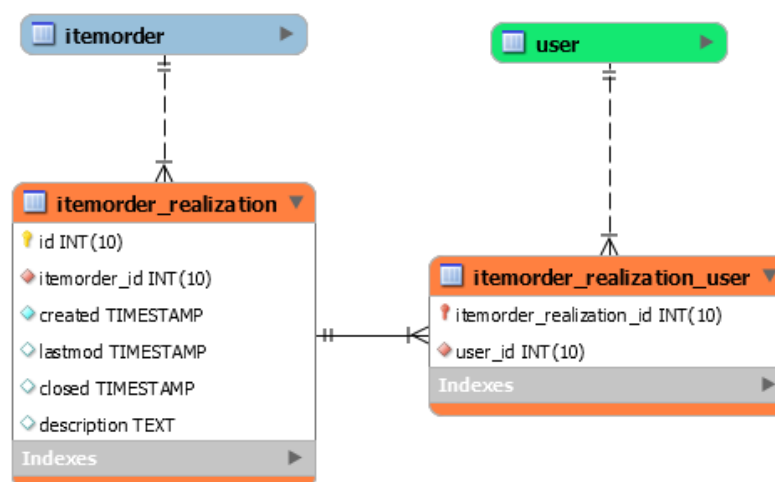
¹³ MIME - Multipurpose Internet Mail Extensions

Tabulka *event* je ve vazbě 1:N s konfiguračními tabulkami dle obrázku (Obr. 40). V konfiguračních tabulkách jsou údaje o barvě, přidružené ikoně a směru vykreslení pro jednotlivé události uvedené v tabulce *event*.



Obr. 40. EER schéma konfiguračních parametrů událostí

Tabulka *itemorder_realization* obsahuje záznamy o provedené realizaci schválené objednávky. Ke každé realizaci je přidělen odpovědný pracovník pomocí vazební tabulky *itemorder_realization_user*. Primárním klíčem v tabulce je atribut *itemorder_realization_id*. Tím je dosaženo, že příslušná realizace bude mít jen jednoho realizátora.



Obr. 41. EER schéma záznamu o realizaci

7 NÁVRH UŽIVATELSKÉHO ROZHRAŇÍ APLIKACE

V průběhu návrhu aplikace jsem vytvořil několik prototypů UI, které jsem modifikoval na základě připomínek uživatelů. Ukázalo se, že jednotliví uživatelé preferují např. různý rozsah informací obsažených na jedné obrazovce. Pro někoho byla nepřehledná forma komplexních formulářů, které obsahovaly veškeré informace na jedné stránce, jiný se zase neztotožnil s rozdělením formuláře do více sekcí. Výsledkem návrhu byl nakonec určitý kompromis. Informace dále nedělitelné, tvořící obsahový či logický celek, jsou sdíleny na jedné obrazovce. Další podpůrné informace jsou odděleny do sekcí a zpravidla zobrazovány na jednotlivých záložkách, vytvořených pomocí Bootstrap stylu `class="nav-tabs"`. Výhodou je tedy určité vyprázdnění a zpřehlednění vykreslovaného prostoru. Zároveň potřebná data jsou načtena jako součást jednoho View a jsou tak okamžitě k dispozici pouhou změnou fokusu na zvolenou záložku.

Snahou bylo vytvořit uživatelské prostředí, které by bylo uživatelsky intuitivní, přehledné a aby plnilo všechny požadované funkce s co nejmenší mírou komplikovanosti i pro méně zdatné uživatele systému. Technickým požadavkem pak byla podpora moderních internetových prohlížečů.

Klasické rozložení uživatelského prostředí představuje hlavička v horní části okna prohlížeče a po jeho levé straně umístěné menu. Zbytek prostoru je určen pro vykreslení vlastního obsahu. Při tvorbě UI jsem využil šablonu SB Admin 2 dostupnou na portále Start Bootstrap [51]. Tato šablona je postavena na komponentách a JS pluginech Twitter Bootstrap a poskytuje širokou paletu elementů uživatelského prostředí.

Ovládání aplikace je realizováno cestou hlavního menu aplikace. Přes toto rozhraní se uživatel dostane ke konkrétní oblasti zájmu. Každá dílčí činnost je zobrazena v samostatném formuláři či okně. V něm se také nachází všechny ostatní ovládací prvky, které souvisí se zvolenou činností. Hlavní menu je tvořeno skupinami, ve kterých jsou seskupeny odkazy na dílčí části aplikace.

Výpis položek, jako je seznam odeslaných objednávek, seznam objednávek ke schválení, apod., je realizován pomocí doplňku Grido. Jedná se o datagrid, který usnadňuje výpis dat, jejich formátování, řazení, stránkování či filtrování [52]. Nabízí jednoduchý způsob použití a přizpůsobení. Na základě připojeného datového modelu je možné zvolit, jaká konkrétní data se vykreslí a co s nimi uživatel může provést. Povolené akce jsou zobrazeny

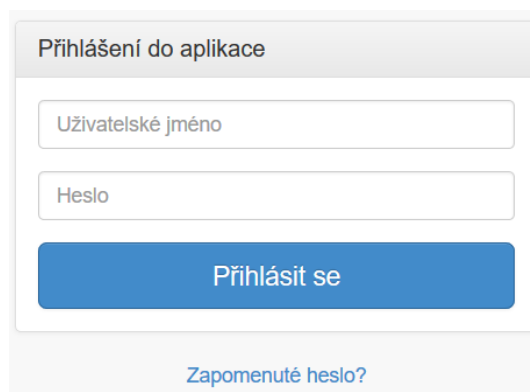
v posledním sloupci tabulky. Jejich rozsah je možné ovlivnit přidělenými uživatelskými právy.

7.1 Přihlášení

Dialog pro přihlášení k aplikaci je vyvolán vždy při přístupu do aplikace nebo v situaci, kdy dojde k automatickému odhlášení uživatele z důvodu dlouhé neaktivity. Je zde využito skutečnosti, že po vytvoření presenteru se vždy volá metoda `startup`. V této metodě je obsažena metoda `checkLogin`, která se stará o ověření stavu přihlášení.

```
1 protected function checkLogin() {
2     if (!$this->getUser()->isLoggedIn()) {
3         $this->redirect('Sign:in', array('backlink'
4             => $this->storeRequest()));}
5 }
```

Každý pokus nepřihlášeného uživatele končí přesměrováním na přihlašovací stránku `../Sign/in`. Uživatel ve formuláři vyplní požadované údaje a po ověření jejich správnosti je přesměrován na hlavní stránku aplikace.



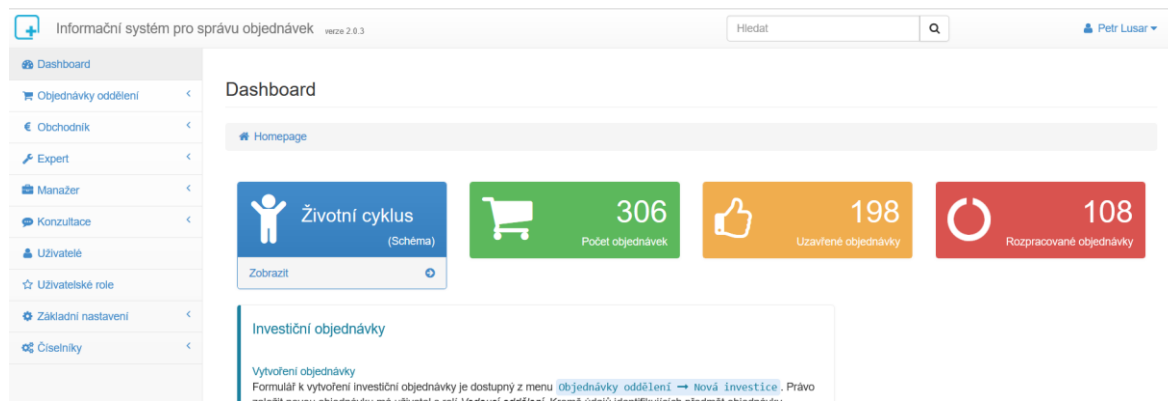
Obr. 42. Dialog přihlášení

V případě, že uživatel zapomene své heslo, má možnost vyžádat jeho obnovu. Dojde k přesměrování na stránku `../User/forgot`. V tomto případě může uživatel bez přihlášení opustit přihlašovací stránku aplikace, aniž by bylo vyžadováno přihlášení do systému. Této výjimky je dosaženo překrytím metody `checkLogin` ve presenteru `UserPresenter`.

```
1 protected function checkLogin() {
2     parent::checkLogin();
3     if ($this->getAction() === 'forgot' || $this->getAction() === 'reset') {
4         return;}
5 }
```

7.2 Hlavní strana

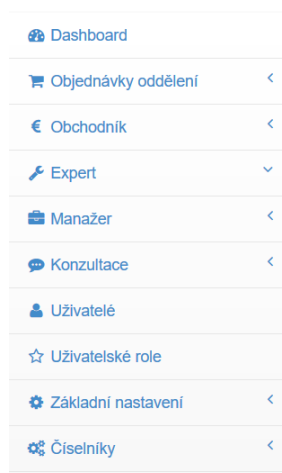
Hlavní strana aplikace poskytuje základní statistiku o počtu zpracovávaných objednávek. Je také využita k předávání sdělení uživatelům, např. o změnách v aplikaci. Tato stránka je společná všem uživatelským rolím.



Obr. 43. Hlavní strana aplikace

7.3 Hlavní menu

Hlavní menu aplikace je v souladu s návrhem rozložení umístěno v levé části okna. Jedná se o hlavní navigační prvek aplikace a je zobrazováno na všech stránkách jako trvalá součást layoutu. Členění položek menu je zpravidla provedeno po skupinách pojmenovaných dle rolí uživatele (Obr. 44). Skupiny lze rozbalit kliknutím na symbol šipky. Zobrazené podmenu je tvořeno odkazy na jednotlivé stránky. Kde je to účelné, je využita struktura víceúrovňového menu. Informace o tom, na které pozici menu se uživatel nachází, je vyjádřena změnou barvy pozadí zvolené položky. Obsahová stránka menu je dynamicky přizpůsobena uživatelské roli přihlášeného uživatele.

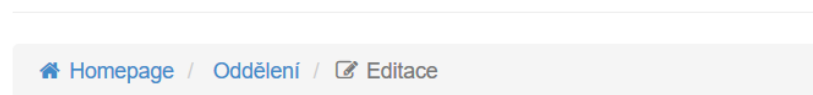


Obr. 44. Hlavní menu

7.4 Drobečková navigace

Pro zobrazení aktuální pozice v aplikaci je využita drobečková navigace. Tento prvek je v šabloně reprezentován CSS třídou `class="breadcrumb"`. Jednotlivé položky tvoří odkazy na příslušné nadřazené části aplikace, podle aktuálního postavení v aplikaci.

Editace záznamu



Obr. 45. Drobečková navigace

7.5 Datagrid Grido

Ve všech případech výpisu seznamu objednávek je použit datagrid Grido. Datagrid je tvořen tabulkou s řádky záznamů a akcí, které jsou pro daný typ záznamu a uživatele povoleny. Součástí je také filtr, který slouží k vyhledávání záznamů podle zadaných kritérií. Tvoří jej formulářové prvky `TextBox`, `ComboBox`, `CheckBox`. Datagrid má k dispozici metodu `setRememberState(TRUE)`, pomocí které lze ukládat stav datagridu do session. To je výhodné pokud z datagridu odskočíme na detail záznamu a po návratu do datagridu potřebujeme zobrazit jeho původní stav (je zachováno filtrování). Pokud toto chování nevyžadujeme, nastavíme jako parametr uvedené metody `FALSE`. Datagrid je poměrně dobře přizpůsobitelný. Umožňuje formátování až na úroveň buňky. Je možné využít pestrou paletu vestavěných metod nebo si vytvořit vlastní volání callback metody [53].

ID	Předmět	Vytvořeno	NS	Název	Typ	Realizace		
185	monitor počítače	14.09.2015	1123	CHIR lůžka C	zboží	✓	✓	Zobrazit
184	Židle k psacímu stolu	14.09.2015	2100	INT řídící část	zboží	✗	✗	Zobrazit
183	Židle k psacímu stolu	14.09.2015	2111	INT ambulance	zboží	✓	✓	Zobrazit
182	Lednice	14.09.2015	2112	INT ambulance (Pol)	zboží	✓	✓	Zobrazit
181	Sprchovací léhátko (lůžko)	11.09.2015	1123	CHIR lůžka C	zboží	✗	✗	Zobrazit
180	Klimatizační zařízení	11.09.2015	2421	PED lůžka novorozenci	zboží	✗	✗	Zobrazit
179	vozik na špinavé prádlo	10.09.2015	2921	Společný lůžkový fond	zboží	✓	✓	Zobrazit
178	převozový vozík sedačka	10.09.2015	2921	Společný lůžkový fond	zboží	✓	✓	Zobrazit
177	vyšetřovací stůl	09.09.2015	2422	PED lůžka kojenci, větší děti+JIP	zboží	✓	✓	Zobrazit
176	Infuzní stojany	09.09.2015	2422	PED lůžka kojenci, větší děti+JIP	zboží	✓	✓	Zobrazit

Jen schválené objednávky Jen nerealizované objednávky

← Předchozí 1 ... 9 ... 11 12 13 14 15 16 17 ... 24 ... 31 Další →

Položky 131 - 140 z 306 10

✗ Neprovedeno 🔄 Provedeno pro část položek objednávky ✓ Provedeno pro všechny položky objednávky

Obr. 46. Výpis seznamu objednávek v datagridu

7.6 Záznam objednávky

Zobrazení konkrétní objednávky je dostupné z nadřazeného datagridu volbou akce *Zobrazit*. Na stránce detailu záznamu se zobrazují údaje objednávky uspořádané do záložek (Obr. 47). Na hlavním panelu jsou k dispozici základní údaje identifikující předmět objednávky.


Základní údaje	
Předmět objednávky:	Notebook
Nákladové středisko:	6891
Oddělení:	Oddělení informačních systémů
Typ objednávky:	zboží
Způsob realizace:	z prostředků Vsetinské nemocnice a.s.
Očekávaná cena:	35 000 Kč

Obr. 47. Ukázka záznamu objednávky

Konkrétní položky objednávky může uživatel zobrazit na záložce *Položky*. Protože z procesního hlediska musí být odborně posuzována každá položka zvlášť, tvoří každá z nich odkaz na samostatnou stránku, která obsahuje detail položky (viz 7.7 - Detail položky objednávky).

Název položky	Počet	Cena
Dell Inspiron 15	1	0

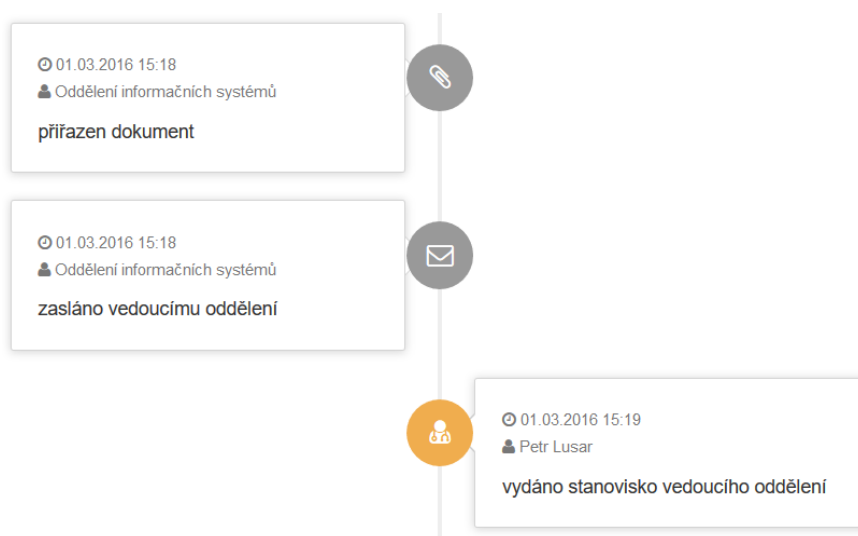
Obr. 48. Výpis položek objednávky

Záložka *Dokumenty* obsahuje tabulkový seznam dokumentů, které byly přiřazeny k objednávce (Obr. 49). Jednotlivé řádky tvoří hypertextový odkaz, který po kliknutí otevře zvolený dokument v asociovaném programu. Aplikace umožňuje editovat název dokumentu tlačítkem . Kdykoliv v průběhu schvalovacího procesu je možné vkládat související dokumentaci. Oprávnění je uděleno každému uživateli s právem přístupu k objednávce.

Název souboru	Velikost souboru	Čas uložení	Autor
dell15.jpg	34.96 kB	01.03.2016 15:18:05	Oddělení informačních systémů

Obr. 49. Výpis dokumentů přiřazených k objednávce

Záznam průběhu zpracování objednávky je zobrazen v samostatné záložce. Častým dotazem uživatelů byla otázka, kde se právě nachází jejich požadavek. Smyslem tohoto prvku tedy bylo nabídnout uživateli požadovanou odpověď. Informace se zobrazuje pomocí grafické časové osy, na níž se chronologicky „přilepují“ jednotlivé události tak jak proběhly (Obr. 50). Ikona, barva a směr vykreslení grafického prvku je určen typem události, jak jsem definoval již v EER modelu. Součástí sdělení jsou data o tom kdo, v jakém čase a jakou událost v aplikaci nad příslušnou objednávkou vyvolal. Tento prvek tak plní zároveň i funkci omezeného logování předem vymezených dějů. V průběhu složitějšího průběhu schvalovacího řízení, kdy je objednávka opakovaně vracena k doplnění, může narůstat i rozsah přidružených událostí. Někteří uživatelé reportovali ztrátu přehlednosti. Za tímto účelem vznikl další prvek, který vykresluje postup schvalovacího procesu na základě hodnoty atributů $\langle prefix \rangle flag$ z tabulky *order*. Tento prvek je ve zmenšené verzi dostupný na hlavním panelu záznamu objednávky nebo je ve své větší variantě součástí stránky vykreslující detail položky objednávky (Obr. 51).



Obr. 50. Grafický timeline průběhu zpracování

7.7 Detail položky objednávky

Zcela v horní části vykreslené stránky je umístěn horizontální ukazatel průběhu stavu vyřizování objednávky. Dále jsou zobrazeny základní údaje objednávky, informace o přiřazených odborných oblastech a vypracované odborné posudky. K položce je možné v libovolné fázi zpracování vkládat komentáře.

Homepage / Objednávka 6.316 Notebook / Náhled

- Požadavek z oddělení
- Stanovisko vedoucího oddělení
- Přiřazení odborné oblasti
- Odborné posudky
- Stanovisko vedoucího CN
- Stanovisko náměstka HTS
- Ukončení schvalovacího řízení
- Převzetí k realizaci
- Ukončení realizace

Základní údaje

Nákladové středisko:	6891
Oddělení:	Oddělení informačních systémů
Název položky:	Dell Inspiron 15
Počet:	1
Celková cena (s DPH):	35 700 Kč
Upřesnění požadavku:	Původní notebook je pro neopravitelnou poruchu navržen k vyřazení. Jedná se o náhradu.

Zařazeno do oblastí

Výpočetní technika

Přiřadit oblast

Komentáře

Vložte komentář ... **Odeslat komentář**

Obr. 51. Detail položky objednávky

Uživatel s příslušnou kompetencí může k položce objednávky přiřadit odbornou oblast, ve které bude experty posuzována. Pomocí tlačítka *Přiřadit oblast* zobrazí formulář pro výběr odborných oblastí, provede požadovaný výběr a uloží (Obr. 52).

Zařazení položky do oblasti

Oblast:

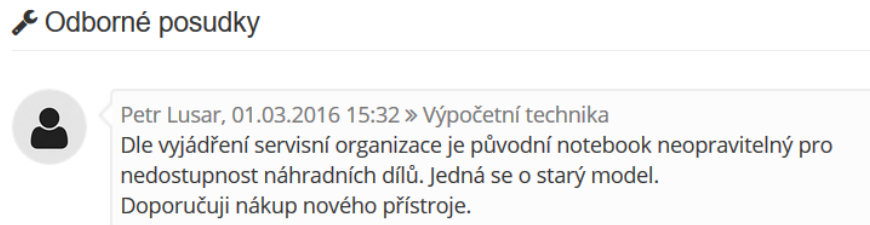
- Automobilová technika
- Kvalita (QM)
- Lékařská péče
- MTZ
- Ošetrovatelská péče
- Personální a mzdová agenda
- Stavební projekt
- SZM
- Technicko - provozní zabezpečení
- Výpočetní technika
- Zdravotní pojišťovny
- Zdravotnická technika

click jednoduchý výběr
Ctrl + click vícenásobný položkový výběr
Shift + click vícenásobný souvislý výběr

Zařadit **Zavřít**

Obr. 52. Formulář pro výběr odborných oblastí

Jednotlivé vypracované posudky, jak již bylo uvedeno, jsou součástí stránky detailu položky objednávky, ke které byly vypracovány (Obr. 53). Kromě samotného odborného vyjádření nesou i informaci kdo posuzoval, kdy a pro jakou oblast.



Obr. 53. Vložený odborný posudek

Grafický prvek může zobrazovat i avatara¹⁴ autora posudku. Podmínkou je, že obrázek formátu JPG je uložen v adresářové struktuře webu. V šabloně je pak voláno

```
1 
```

Pomocí filtru `|checkImage` se vyhodnotí dostupnost autorova avatara a podle návratové hodnoty se vykreslí příslušný obrázek. V `BasePresenteru`, ze kterého dědí všechny presentery aplikace, je umístěna funkce

```
1 public function checkImage($name) {
2     $path = 'img/avatar/';
3     $info = @getimagesize($path . $name);
4     if ($info) {
5         return $path . $name; }
6     return $path . 'NA.jpg';
7 }
```

a vytvořený filtr je zaregistrován

```
1 public function createTemplate($class = NULL) {
2     $tpl = parent::createTemplate($class);
3     $tpl-&#x27;registerHelper('checkImage', callback($this, 'checkImage'));
4     return $tpl;
5 }
```

¹⁴ Reprezentuje určitého uživatele. Většinou se jedná o obrázek.

8 NÁVRH STRUKTURY APLIKACE

System bude rozdělen do několika vrstev, jak vyplývá z použití MVP architektury Nette Framework. Tento postup zajistí nezávislost jednotlivých vrstev a možnost sdílení komponent v rámci aplikace.

8.1 Modelová vrstva

Samotnou manipulaci s daty provádí vrstva Model. Tato vrstva je tvořena sadou objektů umístěných v adresáři *Model* kořenového adresáře aplikace. Jednotlivé repositáře Modelu jsou pojmenovány *<jmeno_modelu>Repository* a dědí od abstraktního objektu *Repository*. V tomto objektu jsou definovány všechny společné metody pro práci s daty. Modelová vrstva nicméně obsahuje i část aplikační logiky. Řada metod tak nabízí kromě zmíněné manipulace s daty i jejich interpretaci, popřípadě zpracování.

Pro připojení k databázi je použita třída `Nette\Database\Connection`. Přístup k datům je realizován pomocí třídy `Nette\Database\Context`. Pomocí této třídy lze pokládat také dotazy, voláním metody `query`. Toho jsem využil pouze ve specifických případech dotazů. Jednou z typických situací bylo použití klíčového slova `UNION` v SQL dotazu, který pomocí `Nette Database Selection`, resp. `DiscoveredConventions` nebyl dobře interpretován.

```
1 function getGridSelection(){
2     return $this->connection->query('(SELECT 1 AS sort_col, id, text,
3         valid FROM justify WHERE id != 1) UNION
4         (SELECT 2, id, text, valid FROM justify
5         WHERE id = 1) '
6         . 'ORDER BY sort_col,text')->fetchAll();
7 }
```

Třída `Nette\Database\Table` poskytuje instanci `Selection`. Pomocí metody `getTable` vrátím klonovanou instanci objektu proměnné `$table`, která je uložena právě v instanci `Nette\Database\Table\Selection`. Tato instance umožňuje filtrování dat a vrací je jako `ActiveRow`. Jednoduché podmínky filtrování jsou vytvořeny voláním metody `where` s libovolným počtem argumentů. Vícenásobné volání metody `where` spojí podmínky pomocí operátoru `AND`. Následující volání jsou shodná:

```
1 $selection->where('id', 1)->where('name', $name);
2 $selection->where('id = ? AND name = ?', 1, $name);
```

8.2 Řídící vrstva

Řídící vrstva aplikace je tvořena presentery. Jak bylo uvedeno v teoretické části, presentery zprostředkovávají komunikaci mezi modelem a vrstvou View, tedy řídí běh aplikace. Všechny presentery aplikace jsou vytvořeny jako třída, která dědí od `Net-te\Application\UI\Presenter` a jsou umístěny v adresáři *Presenters* kořenového adresáře aplikace. Každý objekt Modelu má vytvořenu vlastní třídu presenteru s názvem `<jmeno_modelu>Presenter`. Všechny tyto třídy mají společného abstraktního předka *BasePresenter*, v němž je definována základní společná funkcionalita. Metody uvedené v abstraktní třídě je možné díky překrytí upravit a zajistit tak potřebnou změnu funkcionality. Příkladem je společná metoda `startup`, resp. `checkLogin`, které zajišťují kontrolu přihlášeného uživatele a ověření jeho uživatelských práv pro vykonání požadované akce (str. 69).

9 ZABEZPEČENÍ

Zabezpečení webové aplikace před různými druhy zranitelností je integrální součástí frameworku Nette. Tato vlastnost byla popsána v teoretické části. Budovaný informační systém je nicméně potřeba zabezpečit také na úrovni přístupu jednotlivých uživatelů systému. K tomu je v Nette k dispozici předpřipravená implementace autorizátoru ze třídy `Nette\Security\Permission`.

9.1 Řízení uživatelských oprávnění

Cílem řízení přístupu uživatele k informačnímu systému, resp. k jeho obsahu, je chránit informace během jejich vstupu, zpracování, uložení, přenosu a výstupu proti ztrátě dostupnosti, integrity a důvěrnosti a při jejich likvidaci proti ztrátě důvěrnosti [54]. K rozlišení kompetencí uživatelů v informačním systému využívám přiřazení předdefinovaných uživatelských rolí podle skupin jednotlivých uživatelů. Pro práci s rolmi jsou v Nette Framework dostupné funkce poskytující podporu, kterou požadujeme pro zabezpečení aplikace před neoprávněným přístupem a manipulací s daty.

Autentizace je poskytována ve třídě `UserManager` implementací rozhraní `Nette\Security\IA Authenticator`. Tato třída má jedinou metodu `authenticate`. Jejím úkolem je vrátit identitu uživatele. V případě neúspěchu je vyvolána výjimka `Nette\Security\AuthenticationException`. V případě úspěšného přihlášení vrací metoda `authenticate` identitu uživatele jako objekt, který je instancí třídy `Identity`, implementující rozhraní `Nette\Security\IIdentity`. Instanci této třídy byly předány při vytvoření uživatelské role. Tyto role jsou následně využity v procesu autorizace uživatele.

V řešení Nette je to autorizátor, který ověřuje oprávnění uživatele. Tento objekt je instancí třídy `Nette\Security\Permission`, která implementuje rozhraní `Nette\Security\IAuthorizator`. Toto rozhraní má jedinou metodu `isAllowed` jejímž úkolem je rozhodnout, zda má daná role povolení provést příslušnou operaci s určitým zdrojem.

Nette Framework má k dispozici předpřipravenou implementaci autorizátoru, která poskytuje programátorovi ACL¹⁵ vrstvu pro řízení práv a přístupů. Pro práci s ní je potřeba definovat role, zdroje a činnosti. Role a zdroje je možné vytvářet hierarchicky. Zároveň lze využít dědičnosti rolí.

V informačním systému rozlišuji tyto role:

- Uživatel bez oprávnění (guest),
- Oddělení (department),
- Vedoucí oddělení (chief),
- Odborník (expert),
- Obchodník (trader),
- Manažer (manager),
- Konzultant (consultant),
- Správce bez omezení uživatelských oprávnění (admin).

Zdroje v informačním systému jsem rozdělil podle oblastí tak, aby mohla být volitelně zpřístupněna omezená část aplikace. Oblasti jsou následující:

- Objednávka (order),
- Dokument (document),
- Číselník (catalog),
- Nastavení (setup),
- Konzultace (consultation),
- Uživatelé systému (user).

Nad těmito oblastmi lze provádět tyto činnosti:

- Prohlížení záznamu (view),
- Vytvoření nového záznamu (create),
- Editace záznamu (edit),
- Posouzení záznamu (validate).

¹⁵ ACL – Access Control List

V konfiguračním souboru aplikace je na základě uvedených typů rolí, zdrojů a činností umístěn následující sestavený kód. Tento kód je třídou *Nette\Configurator* převeden do podoby systémového kontejneru.

```
1  authorizator:
2      class: Nette\Security\Permission
3      setup:
4          - addRole('guest')
5          - addRole('department', 'guest')
6          - addRole('chief', 'guest')
7          - addRole('expert', 'guest')
8          - addRole('manager', 'guest')
9          - addRole('trader', 'guest')
10         - addRole('consultant', 'guest')
11         - addRole('admin')
12         - addResource('order')
13         - addResource('document')
14         - addResource('setup')
15         - addResource('user')
16         - addResource('catalog')
17         - addResource('consultation')
18         - allow('guest', ['user', 'order'], 'view')
19         - allow('department', 'order', 'create')
20         - allow('department', 'document', 'create')
21         - allow('chief', 'order', ['create', 'validate'])
22         - allow('chief', 'document', 'create')
23         - allow('consultant', 'consultation',
24             ['create', 'view'])
25         - allow('expert', 'document', 'create')
26         - allow('trader', 'order', 'validate')
27         - allow('trader', 'document', 'create')
28         - allow('trader', 'setup', 'view')
29         - allow('trader', 'catalog', ['create', 'view'])
30         - allow('manager', 'order', 'validate')
31         - allow('manager', 'document', 'create')
32         - allow('admin', Nette\Security\Permission::ALL,
33             Nette\Security\Permission::ALL)
```

9.2 Bezpečné uložení hesla

Proces ověření identity uživatele je založen na porovnání hesla zadaného uživatelem s heslem uloženým v databázi. Z důvodu zvýšení bezpečnosti není heslo nikde v systému uloženo v otevřené podobě. K zabezpečení hesla je použita jeho konverze na vypočítaný hash. Tato metoda přirozeně funguje i jako ochrana před zneužitím ze strany administrátora systému.

Pro transformaci hesla před jeho uložením do databáze je použita PHP funkce `SHA1()`, která jako vstupní parametr přijímá řetězec znaků a návratovou hodnotu je potom 40 znaků hash otisku. Při ověřování identity uživatele tak není porovnáváno heslo uživatele, ale jeho hash otisk. Pro zvýšení zabezpečení je k heslu každého uživatele přidáván náhodný řetězec znaků. Tato technika se označuje jako přidávání soli. Výhodou je zvýšení složitosti hesla i v případě, že uživatel zvolil triviální heslo.

ZÁVĚR

V rámci této práce byl navržen a implementován informační systém pro podporu řízení schvalovacího procesu objednávek služeb a zboží ve Vsetínské nemocnici a.s. K dosažení tohoto úkolu bylo nutné se podrobně seznámit s pracovním postupem tohoto procesu, tak jak je definován vnitropodnikovou směrnicí a jak byl charakterizován v diskuzi s jednotlivými účastníky schvalovacího procesu. Na základě těchto informací byl sestaven diagram podnikového procesu, který byl jedním ze stavebních kamenů pro definování případů užití. Dalším důležitým zdrojem pro návrh informačního systému bylo formulování funkčních a nefunkčních požadavků na informační systém. Popsaný podnikový proces a přehled požadavků co musí aplikace splňovat, byly nezbytným základem pro sestavení diagramů případů užití. V následující fázi jsem mohl přistoupit k pojmenování všech objektů, které budou v informačním systému použity. Tyto objekty jsem použil k sestavení entitně relačního modelu databáze. Ze sestaveného modelu již byly patrné všechny entity a relace, definované mezi nimi. Rozšířením ER modelu o atributy, primární klíče a cizí klíče vznikl rozšířený entitně relační model databáze. EER model obsahoval již také relace mezi entitami za pomoci vazebních tabulek. Jakmile byla známá konečná podoba databáze, mohl jsem přistoupit k návrhu aplikační vrstvy informačního systému.

Informační systém byl implementován ve formátu webové aplikace za použití kombinace skriptovacího jazyka PHP a databázového serveru MySQL. Tato kombinace je v současnosti nejpoužívanější. Díky politice volně šiřitelného software zároveň vyhovuje požadavku zadavatele na nulové poplatky za softwarové licence. Další výhodou zvoleného konceptu je, že požadavky na výkon a programové vybavení klientských stanic nejsou nijak vysoké. Jako dostačující je běžné kancelářské PC s instalovaným internetovým prohlížečem a připojením k síti LAN. Uživatelské rozhraní webové aplikace bylo vytvořeno s využitím šablony a za použití frameworku Twitter Bootstrap a skriptovacího frameworku jQuery. Důraz jsem kladl především na jednoduchost, intuitivnost a přehlednost prostředí.

Vytvořený informační systém zajišťuje požadovanou obsluhu procesu schvalování objednávek služeb a zboží a implementuje všechny požadavky, které byly vymezeny v průběhu vývoje. V rámci vývoje byla aplikace naplněna zkušebními daty a byla ověřena její funkčnost.

Informační systém byl nasazen do zkušebního provozu ve Vsetínské nemocnici a.s. v průběhu května 2015. Během dvouměsíčního zkušebního provozu byl systém otestován

přímo koncovými uživateli. Na základě jejich podnětů byly opraveny některé drobné nedostatky. V některých částech došlo také k úpravě vzhledu a ovládání aplikace tak, aby její použití bylo pro uživatele komfortnější a intuitivnější.

V současné době ve Vsetínské nemocnici a.s. probíhá přibližně 10 měsíců ostrý provoz vybudovaného informačního systému. Během této doby bylo v systému zpracováno cca 410 případů objednávek. Zejména na základě zpětné vazby od uživatelů na všech úrovních zpracování je možné konstatovat, že systém řeší zpracovávanou problematiku efektivně a ke spokojenosti uživatelů. Přínosem pro firmu je pak zpřehlednění celého procesu schvalování objednávek a schopnost řídit nákupy zboží či služeb přehledně a v kontextu s požadavky ostatních oddělení.

Požadovaný úkol tvorby informačního systému pro řízení procesu objednávek služeb a zboží ve Vsetínské nemocnici a.s. se podle mého názoru podařilo úspěšně naplnit.

Během vývoje aplikace jsem se neseťkal s žádným problémem, v jehož důsledku bych se musel odklonit od původního konceptu, nebo který by bránil jeho úspěšnému dokončení. Podařilo se především dodržet modulární koncept v architektuře MVP. Díky tomuto řešení lze v budoucnu reagovat na případné nové požadavky a zajistit úpravu či vybudování nových funkcí informačního systému.

SEZNAM POUŽITÉ LITERATURY

- [1] Co je a není CRM. *Systemonline* [online]. 2012 [cit. 2016-02-21]. Dostupné z: <http://www.systemonline.cz/crm/co-je-a-neni-crm.htm>
- [2] Co je SharePoint? *Office* [online]. Microsoft, 2016 [cit. 2016-02-21]. Dostupné z: <https://support.office.com/cs-cz/article/Co-je-SharePoint-97b915e6-651b-43b2-827d-fb25777f446f>
- [3] Není SharePoint jako SharePoint. *Živě.cz* [online]. Mladá fronta a.s., 2012 [cit. 2016-02-21]. Dostupné z: <http://www.zive.cz/clanky/neni-sharepoint-jako-sharepoint/sc-3-a-163574/>
- [4] Licencování SharePointu 2013. *Office* [online]. Microsoft, 2016 [cit. 2016-02-21]. Dostupné z: <https://products.office.com/cs-cz/sharepoint/sharepoint-licensing-overview>
- [5] The Best PHP Framework for 2015: SitePoint Survey Results. *Sitepoint* [online]. 2015 [cit. 2016-02-21]. Dostupné z: <http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/>
- [6] Technologie pro tvorbu webových aplikací – 3. díl (PHP, PERL, ASP.NET). *Posterus.sk* [online]. 2012 [cit. 2016-01-20]. Dostupné z: <http://www.posterus.sk/?p=13523>
- [7] PHP. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2015 [cit. 2016-01-20]. Dostupné z: <https://cs.wikipedia.org/wiki/PHP>
- [8] Relační databázové systémy. *Posterus.sk* [online]. 2012 [cit. 2016-01-20]. Dostupné z: <http://www.posterus.sk/?p=13526>
- [9] An Introduction To MySQL Storage Engines. *Linux.org* [online]. 2013 [cit. 2016-01-20]. Dostupné z: <http://www.linux.org/threads/an-introduction-to-mysql-storage-engines.4220/>
- [10] InnoDB as the Default MySQL Storage Engine. *MySQL* [online]. Oracle, 2016 [cit. 2016-01-20]. Dostupné z: <http://dev.mysql.com/doc/refman/5.5/en/innodb-default-se.html>
- [11] MySQL (39) - typy tabulek v MySQL. *Linuxsoft.cz* [online]. 2005 [cit. 2016-02-08]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=968
- [12] MyISAM. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-02-08]. Dostupné z: <https://cs.wikipedia.org/wiki/MyISAM>

- [13] The MyISAM Storage Engine. *MySQL* [online]. Oracle, 2016 [cit. 2016-02-08]. Dostupné z: <http://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html>
- [14] InnoDB as the Default MySQL Storage Engine. *MySQL* [online]. Oracle, 2016 [cit. 2016-02-09]. Dostupné z: <http://dev.mysql.com/doc/refman/5.7/en/innodb-default-se.html>
- [15] InnoDB. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2014 [cit. 2016-02-09]. Dostupné z: <https://cs.wikipedia.org/wiki/InnoDB>
- [16] ŠVIRGA, Václav. Modelování MySQL databáze pomocí MySQL Workbench. *ABC Linuxu* [online]. Nitemedia s. r. o., 2013 [cit. 2016-02-09]. Dostupné z: <http://www.abclinuxu.cz/clanky/modelovani-mysql-databaze-pomoci-mysql-workbench>
- [17] MySQL Workbench. *MySQL* [online]. Oracle, 2016 [cit. 2016-02-10]. Dostupné z: <https://www.mysql.com/products/workbench/>
- [18] Informační web o nástroji Enterprise Architect. *Enterprise Architect* [online]. 2016 [cit. 2016-02-10]. Dostupné z: <http://www.enterprise-architect.cz/>
- [19] Ultimate Modeling Power. *Enterprise Architect* [online]. Sparx Systems, 2016 [cit. 2016-02-10]. Dostupné z: <http://www.sparxsystems.com.au/products/ea/index.html>
- [20] Úvod do UML. *Itnetwork.cz* [online]. 2015 [cit. 2016-02-10]. Dostupné z: <http://www.itnetwork.cz/navrhove-vzory/uml/uml-uvod-historie-vyznam-a-diagramy/>
- [21] ŠILHAVÝ, Radek. *Softwarové inženýrství: Úvod do softwarového inženýrství*. Zlín, 2016
- [22] BERNARD, Borek. Úvod do architektury MVC. *Zdroják.cz* [online]. 2009 [cit. 2016-02-10]. Dostupné z: <https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/>
- [23] MVC aplikace & presentery. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-09]. Dostupné z: <https://doc.nette.org/cs/2.3/presenters>
- [24] BERNARD, Borek. Prezentační vzory z rodiny MVC. *Zdroják.cz* [online]. 2009 [cit. 2016-02-10]. Dostupné z: <https://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/>
- [25] Model-View-Presenter (MVP). *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-10]. Dostupné z: <https://doc.nette.org/cs/0.9/model-view-presenter>

- [26] Nette Framework. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2015 [cit. 2016-02-08]. Dostupné z: https://cs.wikipedia.org/wiki/Nette_Framework
- [27] Zabezpečení před zranitelnostmi. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-08]. Dostupné z: <https://doc.nette.org/cs/2.3/vulnerability-protection>
- [28] Seznámení s Nette Frameworkem. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-08]. Dostupné z: <https://doc.nette.org/cs/2.3/getting-started>
- [29] Licenční politika. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-08]. Dostupné z: <https://nette.org/cs/license>
- [30] Konfigurace. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-10]. Dostupné z: <https://doc.nette.org/cs/2.3/configuring>
- [31] Dependency Injection. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-10]. Dostupné z: <https://doc.nette.org/cs/2.3/dependency-injection>
- [32] Získávání závislostí. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-10]. Dostupné z: <https://doc.nette.org/cs/2.3/di-usage>
- [33] MATĚJKA, David. *Nette DI cheatsheet* [online]. In: . [cit. 2016-04-14]. Dostupné z: <http://nette.matej21.cz/cs/di>
- [34] Databáze. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-13]. Dostupné z: <https://doc.nette.org/cs/2.3/database>
- [35] ODBC - přehled Open Database Connectivity. *Microsoft* [online]. 2016 [cit. 2016-02-13]. Dostupné z: <https://support.microsoft.com/cs-cz/kb/110093>
- [36] PDO a další novinky v PHP 5.1. *Root.cz* [online]. 2005 [cit. 2016-02-13]. Dostupné z: <http://www.root.cz/clanky/pdo-php-5-1/>
- [37] VRÁNA, Jakub. Databáze v PHP elegantně s NotORM. *Zdroják.cz* [online]. 2010 [cit. 2016-02-13]. Dostupné z: <https://www.zdrojak.cz/clanky/databaze-v-php-elegantne-s-notorm/>
- [38] GRUDL, David. Dibi vs. Nette Database story. *phpFashion* [online]. 2014 [cit. 2016-02-13]. Dostupné z: <https://phpfashion.com/dibi-vs-nette-database-story>
- [39] Database\Table. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-13]. Dostupné z: <https://doc.nette.org/cs/2.3/database-table>
- [40] Database: Selection. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-13]. Dostupné z: <https://doc.nette.org/cs/2.3/database-selection>

- [41] Database: ActiveRecord. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-14]. Dostupné z: <https://doc.nette.org/cs/2.3/database-activerow>
- [42] Šablony. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-14]. Dostupné z: <https://doc.nette.org/cs/2.3/templating>
- [43] Formuláře. *Nette.org* [online]. Nette Foundation, 2016 [cit. 2016-02-15]. Dostupné z: <https://doc.nette.org/cs/2.3/forms>
- [44] Twitter Bootstrap. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-02-17]. Dostupné z: https://cs.wikipedia.org/wiki/Twitter_Bootstrap
- [45] K čemu je dobrý Bootstrap a frontend frameworky? *Zdroják.cz* [online]. 2016 [cit. 2016-02-17]. Dostupné z: <https://www.zdrojak.cz/clanky/k-cemu-je-dobry-bootstrap-frontend-frameworky/>
- [46] JQuery. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-02-17]. Dostupné z: <https://cs.wikipedia.org/wiki/JQuery>
- [47] JQuery UI. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-02-17]. Dostupné z: https://cs.wikipedia.org/wiki/JQuery_UI
- [48] About jQuery UI. *jQueryUI* [online]. 2016 [cit. 2016-02-17]. Dostupné z: <http://jqueryui.com/about/>
- [49] KRAVAL, Ilja. *Analytické modelování informačních systémů pomocí UML v praxi*. 1. vydání. Lipina: Object Consulting, 2010. ISBN 978-80-254-6986-6
- [50] Příklady použití diagramů UML 2.0. *Czweb.org* [online]. 2009 [cit. 2016-04-03]. Dostupné z: http://uml.czweb.org/pripad_uziti.htm
- [51] SB Admin 2. *Start Bootstrap* [online]. 2014 [cit. 2016-03-01]. Dostupné z: <http://startbootstrap.com/template-overviews/sb-admin-2/>
- [52] o5/grido. *GitHub* [online]. 2016 [cit. 2016-03-01]. Dostupné z: <https://github.com/o5/grido>
- [53] Grido: Documentation. *o5.github.io* [online]. 2016 [cit. 2016-03-01]. Dostupné z: <http://o5.github.io/grido-examples/documentation.cs.html>
- [54] JAŠEK, Roman a David MALANÍK. *Bezpečnost informačních systémů*. První. Zlín: Univerzita Tomáše Bati ve Zlíně, 2013. ISBN 978-80-7454-312-8

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ACL	Access Control List
AJAX	Asynchronous JavaScript and XML
BPMN	Business Process Model Notation
CAL	Client Access Licenses
CRM	Customer Relationship Management
CSRF	Cross-Site Request Forgery
DDHM	Dlouhodobý drobný hmotný majetek
DHM	Dlouhodobý hmotný majetek
DI	Dependency Injection
DOM	Document Object Model
DRY	Don't repeat yourself
EER	Enhanced entity–relationship model
ER	Entity-relationship model
FTP	File Transfer Protocol
GPL	General Public License
HTTP	Hypertext Transfer Protocol
IMAP	Internet Message Access Protocol
iOS	Operační systém firmy Apple pro mobilní platformu
KISS	Keep it short and simple
MD5	Message Digest 5
MIME	Multipurpose Internet Mail Extensions
MIT	Massachusetts Institute of Technology
MVC	Model-View-Controller
MVP	Model-View-Presenter
ODBC	Open Database Connectivity

OS X	Operační systém firmy Apple pro počítače Macintosh
PECL	PHP Extension Community Library
PHP	PHP: Hypertext Preprocessor
POP3	Post Office Protocol - Version 3
SEO	Search engine optimization
SHA	Secure Hash Algorithm
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
UML	Unified Modeling Language
XSS	Cross-Site Scripting
UI	User interface

SEZNAM OBRÁZKŮ

Obr. 1. Vzor definice pověřených zaměstnanců pro jednotlivé činnosti zpracování objednávky	13
Obr. 2. Aplikace MySQL Workbench [17]	21
Obr. 3. Aplikace Enterprise Architect [19]	22
Obr. 4. Architektura MVC [22]	23
Obr. 5. Architektura MVP – vzor Passive View [24]	25
Obr. 6. Způsoby předávání závislostí [33]	27
Obr. 7. Životní cyklus presenteru [23]	28
Obr. 8. Schéma databáze	31
Obr. 9. Diagram hlavního toku procesu schvalování objednávky	38
Obr. 10. Diagram hlavního toku procesu realizace schválené objednávky	39
Obr. 11. Diagram aktivity Vedoucího oddělení	40
Obr. 12. Diagram aktivity Experta a Obchodníka	41
Obr. 13. Diagram aktivity Manažera	42
Obr. 14. Vztah generalizace aktérů systému	44
Obr. 15. Speciální aktéři	44
Obr. 16. Základní diagram případů užití administrace	45
Obr. 17. Základní diagram případů užití založení objednávky	46
Obr. 18. Základní diagram případů užití obsluhy objednávky	46
Obr. 19. Základní diagram případů užití oblasti konzultací	47
Obr. 20. Základní diagram případů užití správy číselníků	47
Obr. 21. Diagram případů užití Uživatel	48
Obr. 22. Diagram případů užití Objednávka	51
Obr. 23. Diagram případů užití Stanovisko	53
Obr. 24. Diagram případů užití Konzultace	54
Obr. 25. Diagram případů užití Číselníky	55
Obr. 26. Entita objednávky a položky objednávky	57
Obr. 27. Entita pro zařazení položky do odborné oblasti	58
Obr. 28. Entita pro záznam odborného posudku	58
Obr. 29. Entita pro záznam stanoviska k objednávce	59
Obr. 30. Entita pro dokument a typ dokumentu	59
Obr. 31. Entita pro záznam realizace objednávky	60

Obr. 32. Entita konzultace k objednavce	60
Obr. 33. Entita uživatele a jeho role	61
Obr. 34. Entity vyjadřující organizační členění nemocnice.....	61
Obr. 35. EER schéma uživatelů a odborných oblastí	62
Obr. 36. EER schéma organizačního členění	63
Obr. 37. EER schéma uživatelů a uživatelských rolí.....	64
Obr. 38. EER schéma základních entit	65
Obr. 39. EER schéma přidružených záznamů k objednavce	66
Obr. 40. EER schéma konfiguračních parametrů událostí.....	67
Obr. 41. EER schéma záznamu o realizaci	67
Obr. 42. Dialog přihlášení.....	69
Obr. 43. Hlavní strana aplikace	70
Obr. 44. Hlavní menu	70
Obr. 45. Drobečková navigace	71
Obr. 46. Výpis seznamu objednávek v datagridu	71
Obr. 47. Ukázka záznamu objednávky	72
Obr. 48. Výpis položek objednávky	72
Obr. 49. Výpis dokumentů přiřazených k objednavce.....	73
Obr. 50. Grafický timeline průběhu zpracování	73
Obr. 51. Detail položky objednávky	74
Obr. 52. Formulář pro výběr odborných oblastí	74
Obr. 53. Vložený odborný posudek	75

SEZNAM PŘÍLOH

- P I Listinná verze formuláře objednávky
- P II Celkový rozšířený ER model databáze
- P III BPD Schvalovací řízení
- P IV BPD Realizace
- P V Diagram funkčních požadavků

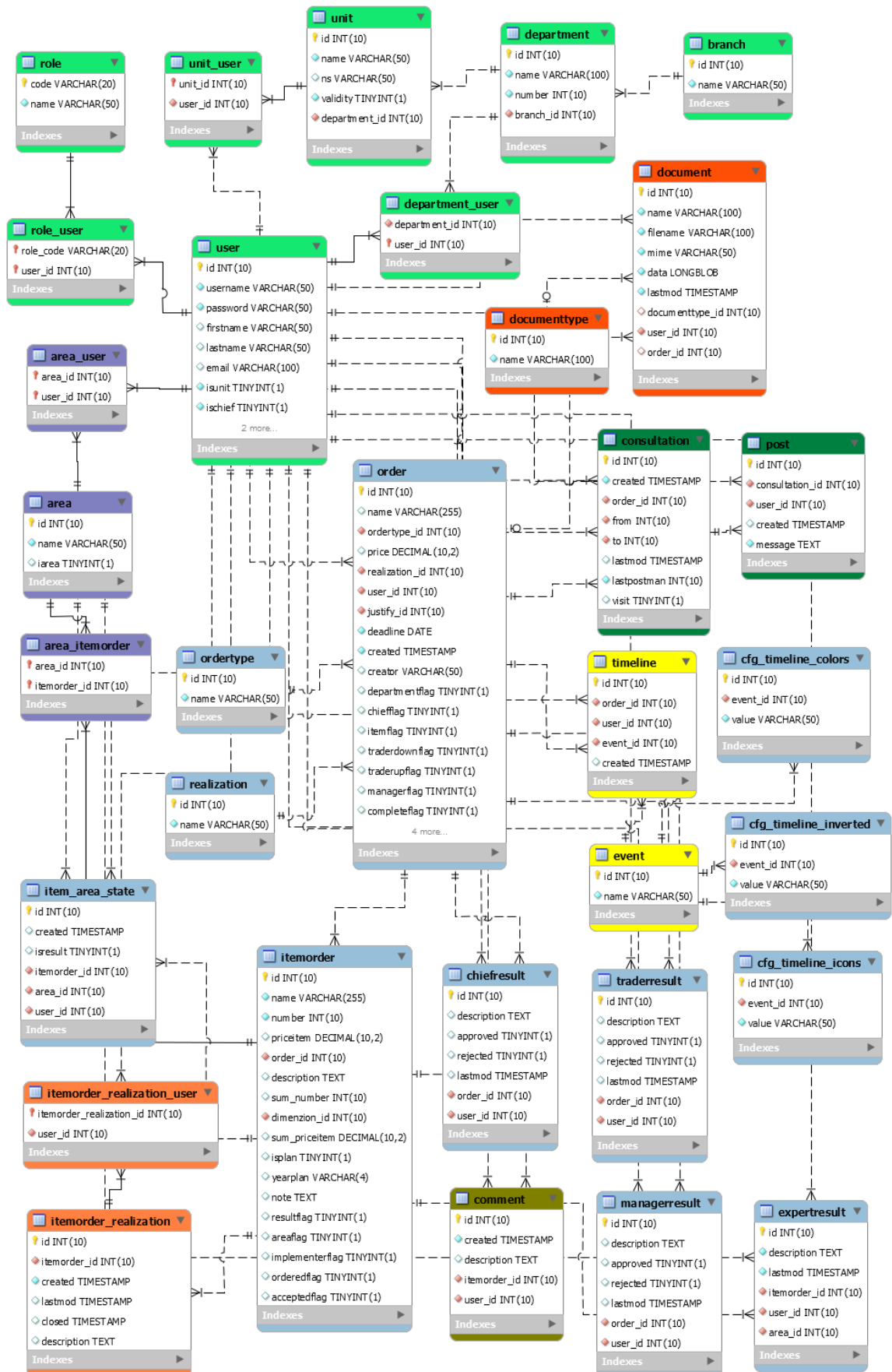
PŘÍLOHA P I: LISTINNÁ VERZE FORMULÁŘE OBJEDNÁVKY

	Objednávka
---	------------

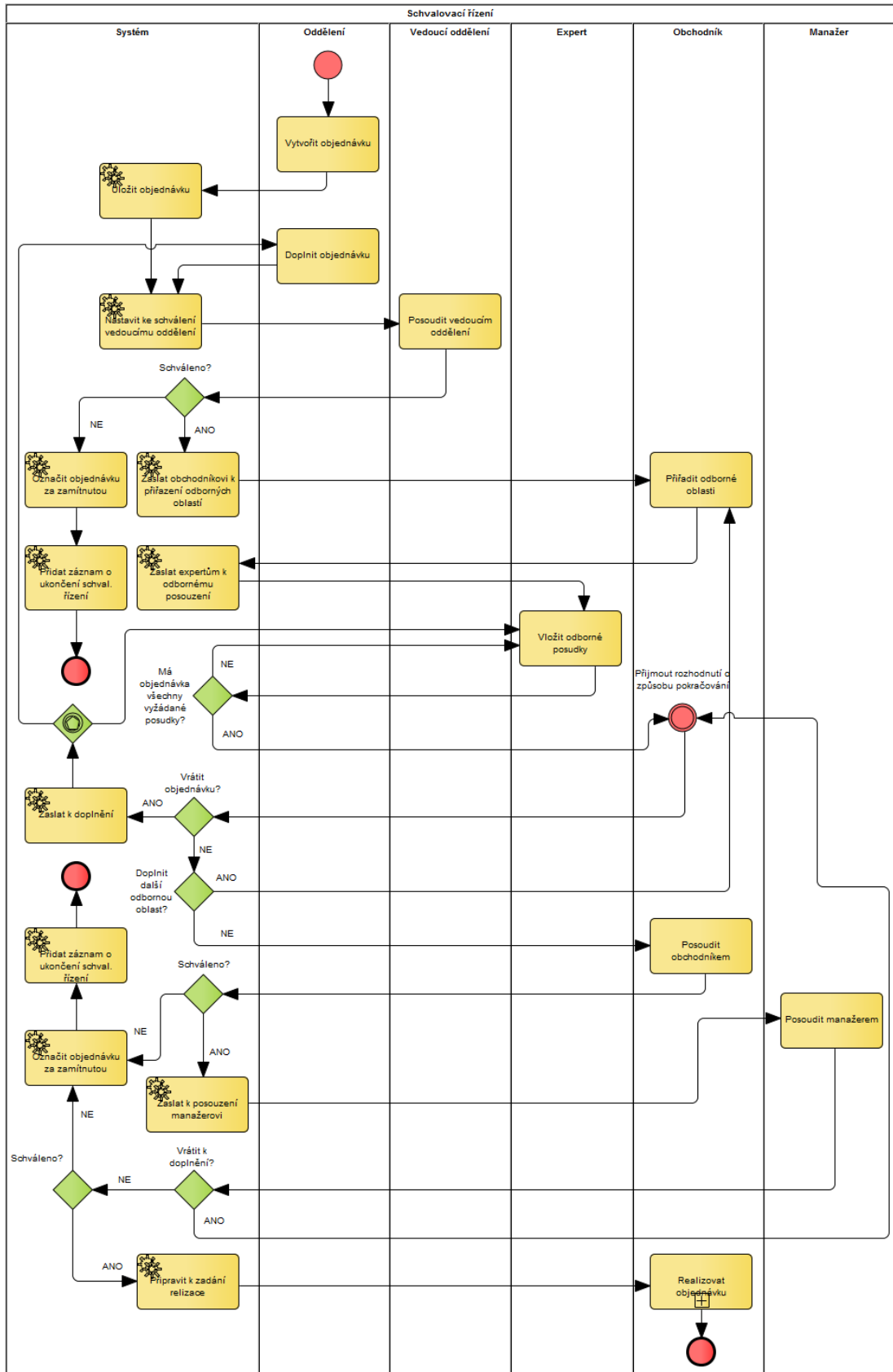
<i>Název:</i>			
zboží		služeb	
Organizační složka <i>(pracoviště předkladatele)</i>			Číslo NS:
Předmět objednávky		Očekávaná cena (vč. DPH): Kč	
V souladu s postupem stanoveným Směrnicí kvality č. SM – 36–0200/09 Nakupování, žádám:			
Realizace: a) vlastními prostředky (dary apod.) b) z prostředků Vsetínské nemocnice a.s.			
Zdůvodnění objednávky ze strany předkladatele:			
Vyjádření odborného pracoviště (OIS, údržby apod.):			
Požadovaný termín realizace:			
<i>Jméno, příjmení a funkce předkladatele</i>	<i>Datum předložení objednávky:</i>	<i>Podpis předkladatele</i>	<i>Podpis vedoucího pracoviště</i>
<i>Tel. kl.</i>			
EOO	Předáno	Doklad o nákupu – nebo Odpověď odd.:	
Přijato:	NVS:		
<i>Vyjádření NVS:</i> <i>Schvaluji – neschvaluji:</i> <i>(zdůvodnění)</i>			<i>Podpis:</i> <i>Dne:</i>

Revize č.: 3 Datum platnosti revize: 12. 4. 2013	Evidenční číslo formuláře F/01/SM 36-0200/09 <hr/> Strana 1/1
---	--

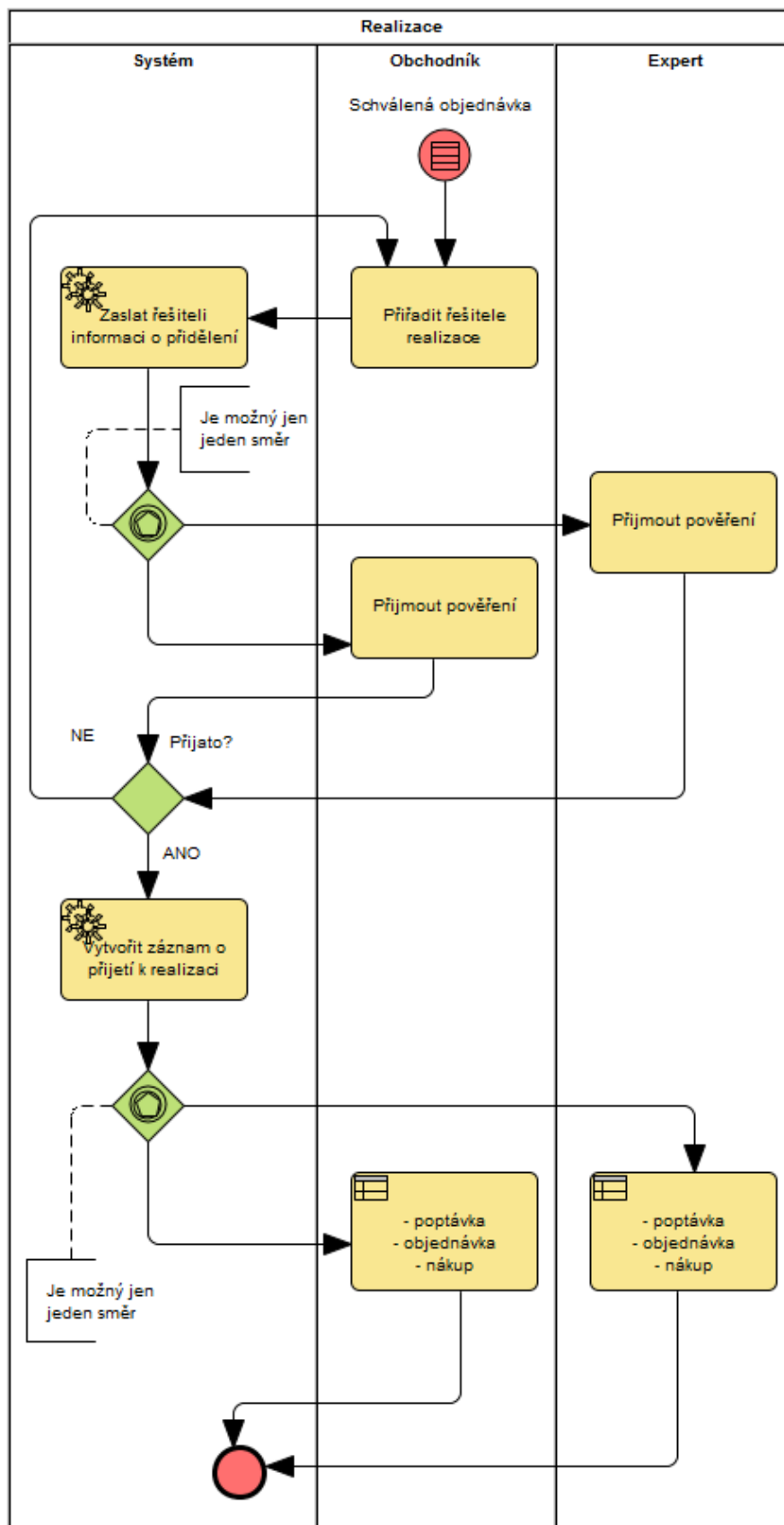
PŘÍLOHA P II: CELKOVÝ ROZŠÍŘENÝ ER MODEL DATABÁZE



PŘÍLOHA P III: BPD SCHVALOVACÍ ŘÍZENÍ



PŘÍLOHA P IV: BPD REALIZACE



PŘÍLOHA P V: DIAGRAM FUNKČNÍCH POŽADAVKŮ

