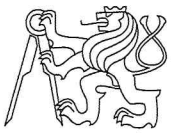
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



# ZADÁNÍ DIPLOMOVÉ PRÁCE

| | |
|---|---|
| **Název:** | Serverová část a integrační vrstva informačního systém pro firmu dodávající IT služby |
| **Student:** | Bc. Filip Tvrdoň |
| **Vedoucí:** | Mgr. Martin Bruoth |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Webové a softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | do konce letního semestru 2015/16 |

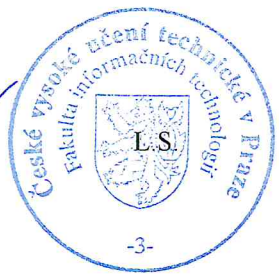
## Pokyny pro vypracování

Pro firmu zabývající se prodejem IT služeb navrhněte a implementujte serverovou část nového IS, který pokryje klíčové business procesy zadavatele. Součástí práce bude také integrační vrstva zajišťující komunikaci s dalšími moduly připojenými k IS (mailový a tiketovací systém). Výsledek bude sloužit jako zdroj dat pro webovou prezentační vrstvu, které bude poskytovat komunikační rozhraní (API).

- Seznamte se s business procesy zadavatele a se současnými používanými IS.
- Seznamte se s platformou Origam.
- Proveďte analýzu a návrh serverové a integrační vrstvy IS na základě dodané specifikace požadavků zadavatele (dodá vedoucí práce).
- Návrh bude respektovat zvyklosti platformou Origam.
- Návrh implementujte, dokumentujte a vhodným způsobem otestujte.
- Součástí práce není návrh prezentační vrstvy IS, pro testování a prezentaci bude použit implicitní frontend generovaný vývojovou platformou Origam.

## Seznam odborné literatury

Dodá vedoucí práce.




Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof.Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 7. února 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

# Serverová část a integrační vrstva informačního systému pro firmu dodávající IT služby

***Bc. Filip Tvrdoň***

Vedoucí práce: Mgr. Martin Bruoth

4. května 2016

# Poděkování

Děkuji svému vedoucímu Mgr. Martinu Bruothovi za vedení diplomové práce, jeho cenné rady, technické vybavené nutné pro implementaci a také zázemí při realizaci. Dále děkuji Tomáši Vavrdovi za technickou konzultaci při návrhu a implementační podporu při realizaci prototypu IS.

# Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na základě níž se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 4. května 2016 .....

### Odkaz na tuto práci

Tvrdoň, Filip. *Serverová část a integrační vrstva informačního systému pro firmu dodávající IT služby.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

# Abstrakt

Diplomová práce se zabývá praktickým použitím techniky „Model Driven Architecture“ a jejího využití při následném vývoji pomocí frameworku Origam. Cílem je analýza a návrh firemního informačního systému integrovaného s dalšími externími komponentami a službami společnosti.

**Klíčová slova** Analýza, Návrh, Model Driven Development, Model Drive Architecture, XML, MDD, MDA, Origam

# Abstract

The master thesis apply practical using of technique „Model Driven Architecture“ and consecutive development in framework Origam. Primary goal is analysis and design of company's information system integrated with other external components and services in company.

**Keywords** Analysis, Design, Model Driven Development, Model Drive Architecture, XML, MDD, MDA, Origam

# Obsah

# Seznam obrázků

# Seznam tabulek

# Úvod

Během poslední dekády jsme ve světě informačních technologií svědky prudkého a stále se zrychlujícího rozvoje. Je nutno podotknout, že v rámci tohoto masivního rozvoje se nejedná pouze o samotný vývoj hardwaru, kde je rychlost vývoje popsána pomocí Moorova zákona, který platí již přes 50 let a ještě několik desetiletí bude podle předpovědí předních technologických firem platit. Velký rozvoj zažívají rovněž softwarové systémy, jejichž pokrok je úzce svázán s rozvíjejícími se technologiemi. Ač jsou tyto obě části IT systémů velmi úzce propojeny, je v jejich vývoji rozdíl. Hardware se snaží jít cestou maximální miniaturizace, energetické efektivity při maximalizaci výkonu a také komplexity zařízení vzhledem k jejich velikosti (maximální množství zařízení v jednom čipu). U softwarových systému vznikají naopak stále větší a komplexnější systémy s větší mírou automatizace a integrace. Ruku v ruce také rostou nároky na výkon hardware, na kterém systémy běží.

Zaměřme se nyní blíže na problematiku informačních systémů a jejich vývoje. Ze strany zadavatele se při specifikaci dají shrnout požadavky do následujících základních bodů:

- Maximum funkcionalit (slovy manažera: „Musí to umět všechno, co si vymyslíme.“).
- Integrace s dalšími či stávajícími systémy.
- Maximální automatizace pro co největší efektivitu práce
- Čas (dodání kompletní aplikace či systému v minimálním čase).
- Náklady na realizaci (maximální poměr cena ku funkcionalitám systému).

Nyní se nabízí otázka, jakým způsobem vést analýzu a vývoj informačního systému, aby návratnost investice nebyla za životností systému a její časový

horizont nebyl v desítkách let. Je takovýto vývoj vůbec možný? Je možné nalézt nějakou techniku vývoje, kterou by to mohlo být uskutečnitelné?

Vývoj informačních systémů se čím dál víc odklání od klasického vývoje pomocí metody vodopádu k moderním agilním metodám či pomocí modelem řízeného vývoje (Model Driven Development). Metoda MDD je známá již od konce 80. let minulého století, ale procento projektů, které ji využívá, není velké. Přínosy vývoje MDD jsou přesto veliké.

## Cíl práce

Cílem této práce je implementace funkčního prototypu informačního systému, který bude sloužit jako podklad pro vývoj v dalších iteracích. Informační systém bude vytvořen na základě specifikace požadavků zákazníka.

Před samotnou implementací bude provedena analýza funkčních požadavků a jejich rozdělení do skupin podle priorit. Bude provedena analýza případů užití s ohledem na obchodní procesy zákazníka. Na základě analýzy proběhne samotný návrh datové vrstvy a návrh aplikační logiky celého informačního systému. Tyto návrhy budou respektovat zvyklosti používání při vývoji informačního systému ve frameworku Origam.

## Motivace

Hlavní motivací zpracování této práce je další rozvoj a rozšíření mých znalostí v oblasti analýzy, návrhu a vývoje aplikací ve frameworku Origam. Díky tomuto projektu s ním pracuji déle než rok a aktuálně se podílím na vývoji dalších systémů v tomto frameworku. V rámci implementace budu vyvíjet nové generické komponenty frameworku Origam pro marketing a prodej softwarových služeb. Také rozšířím integrační část o další komponenty – externí marketingový systém pro hromadného odesílání emailů, SMS bránu a ticketovací systém.

Kapitola 1

# Základy Model Driven Architecture

Následující kapitola přináší obecné a základní informace o technologiích „Model Driven Development“ a „Model Driven Architecture (MDA)“. Zároveň také přibližuje framework Origam použitý pro implementaci prototypu systému. Blíže je zde popsána technická specifikace, jeho možnosti a použité technologie.

## 1.1 Základní principy a pojmy

Na úvod do problematiky je vhodné vysvětlit několik základních pojmů, nutných pro pochopení celé technologie frameworku Origam.

### 1.1.1 Model

Model systému je popis či specifikace daného systému a jeho prostředí vypracovaný pro daný způsob využití. Model je často prezentován jako kombinace textu a grafiky či obrázků. Text může být v modelovacím či přirozeném jazyce. Model může reprezentovat procesy, doménu, software, hardware, prostředí nebo jiné doménově specifické části systému.

### 1.1.2 Pohled

Pohled (anglicky „view“) na systém je technika abstrakce při využití vybrané množiny architektonických konceptů a strukturovaných pravidel se zaměřením na konkrétní části uvnitř daného systému. Pojem „abstrakce“ je zde použit ve smyslu potlačení vybraného detailu směrem k získání zjednodušeného modelu.

### 1.1.3 Platforma

Platforma je množina podsystémů a technologií poskytující ucelenou a kompletní množinu funkcionalit, které jsou vystaveny do okolního světa skrze rozhraní. Jakákoliv aplikace může tyto funkcionality využít bez toho, aby znala jejich způsob implementace v dané platformně. Zjednodušeně se jedná o prostředí, které umožňuje spuštění aplikace. Příkladem platformy je operační systém, Java Virtual Machine a další.

### 1.1.4 Platformní nezávislost

Platformní nezávislost znamená, že model není závislý na konkrétní platformě (též nazýváno platformně nezávislý model – PIM) a používá obecné metody bez znalosti daného systému, například vzdálené volání procedur.

### 1.1.5 Transformace modelu

Transformace modelu je komplexní proces, kterým se převádí vstupní model systému na jiný model daného systému. Součástí procesu je specifikace transformace a vstupní parametry transformace.

### 1.1.6 Model Driven Engineering

Jedná se o obecnou metodiku softwarového vývoje. Klade si za cíl zvýšení úrovně abstrakce specifikace programu/aplikace a zvýšení automatizace vývoje. Hlavní ideou prosazovanou v Model Driven Engineering (MDE) je použití modelů na různých úrovních abstrakce vyvíjeného systému, čímž se zároveň zvyšuje abstrakce specifikace aplikace/systému. Zvýšení automatizace ve vývoji aplikace může být dosaženo pomocí transformací modelu. Model vyšší úrovně může být transformován do modelu nižší úrovně, který může být použit pro vygenerování kódu nebo může být přímo interpretován.

S MDE souvisí další pojem MDA, který slouží jako vize MDE skupiny Object Management Group, která stojí za definicí specifikace MDA.

### 1.1.7 Model Driven Development

Model Driven Development je specifikován jako rychlý a rapidní vývoj komplexních systému na základě informací uložených v modelech, které jsou menší a komplexnější než samotný vyvíjený systém.

Jinými slovy také lze říci, že se jedná se o způsob řešení problémů použitím všudypřítomných modelů a použitím modelů jako vstupu do parametrizovaných generátorů. Pokud bychom v rámci implementace zjistili, že modely jsou větší a komplexnější než problém, který mají řešit, pravděpodobně jsme někde udělali chybu. Jedná se o způsob vývoje softwaru použitelný v obecném přístupu pomocí MDE, který nemusí být omezený pouze na MDA.

### 1.1.8 Model Driven Architecture

Pojem MDA popisuje přístup či koncept vývoje softwaru založený na modelech, jejich transformacích a následném spouštění. Koncept byl definován v roce 2001 sdružením Object Management Group, Inc., v roce 2014 byla vydána druhá verze specifikace. MDA koncept poskytuje přidanou hodnotu k modelům a architektuře během podpory celého jejich životního cyklu, ať už se jedná o fyzické, organizační či informační systémy. MDA koncept reprezentuje a podporuje vše od business požadavků až po implementaci na konkrétní technologii. Využití MDA modelů umožňuje lépe pracovat s velkými a komplexními systémy a také zlepšuje komunikaci a spolupráci mezi organizacemi, lidmi, hardwarem a systémem.

#### 1.1.8.1 Hlavní prvky MDA konceptu

Hlavním rysem MDA, který nám dovoluje pracovat s komplexností systémů a získávat informace z modelů, je definování struktury, sémantiky a notace modelů za pomoci využití průmyslových standardů — modely vyhovující tomuto standardu jsou „MDA modely“. MDA modely mohou být dále použity v produkci jako dokumentace, ověření specifikace, specifikace systému, vytvoření zdrojového kódu či přímo ke spuštění samotného systému.

MDA využívá modely ke zlepšení agility plánování, návrhu a ke zvýšení kvality a údržby výsledného produktu. V MDA lze také modely použít přímo k implementaci výsledného řešení či částí kódu tak, že je možno z modelů vygenerovat spustitelný kód nebo je přímo spustit.

MDA definuje množinu standardů pro vývoj zahrnující reprezentaci a způsob změn v modelu v různých modelovacích jazycích, transformace modelů, vytváření dokumentace a spuštění aplikace na základě modelu. MDA modely mohou popisovat systémy na jakékoliv úrovni abstrakce nebo z různých pohledů, od enterprise architektury až po cílovou implementaci. MDA vytváří spojení mezi různými pohledy a abstrakcí.

V rámci MDA konceptu můžeme mluvit o následujících třech architektonických vrstvách:

a. **Business nebo doménový model** – model zachycující aktuální reálné objekty z dané domény; nejedná se o reprezentaci těchto objektů v rámci systému. V konceptu MDA se tyto modely z historického hlediska nazývají „Computation Independent Model (CIM).“

b. **Logický model systému** – model reprezentující jednotlivé komponenty systému a jejich interakci mezi sebou, lidmi a organizací za účelem splnit vytyčený cíl.

c. **Implementační model** – část systému či subsystému, který je implementován tak, že nám přímo poskytuje určitou funkčnost. Implementace modelu je typicky svázaná s technickou implementací či platformou.

#### 1.1.8.2 Interpretace a transformace MDA modelu

Hlavní přínos konceptu MDA spočívá v automatickém převodu modelu do spustitelného informačního systému. Tento proces převodu pomáhá snižovat náklady, čas a riziko implementace a údržby vyvíjeného informačního systému, zatímco můžeme realizovat jeho další rozvoj.

Existují dva hlavní přístupy pro převod modelu do informačního systému:

1. Aplikací transformační šablony na model, čímž vzniká přímo část spustitelného kódu. Příkladem může být transformace obchodních procesů informačního modelu do XML schématu, C# či Java kódu.

2. Interpretační engine, který je schopen přeložit či spustit model. Jeho realizace je provedena v platformně závislém jazyce. Výsledek je podobný přímému spuštění zdrojového kódu, pouze máme přidanou jednu vrstvu navíc (analogie může být Java Virtual Machine). Engine může například komunikovat s ostatními systémy pomocí XML zpráv.

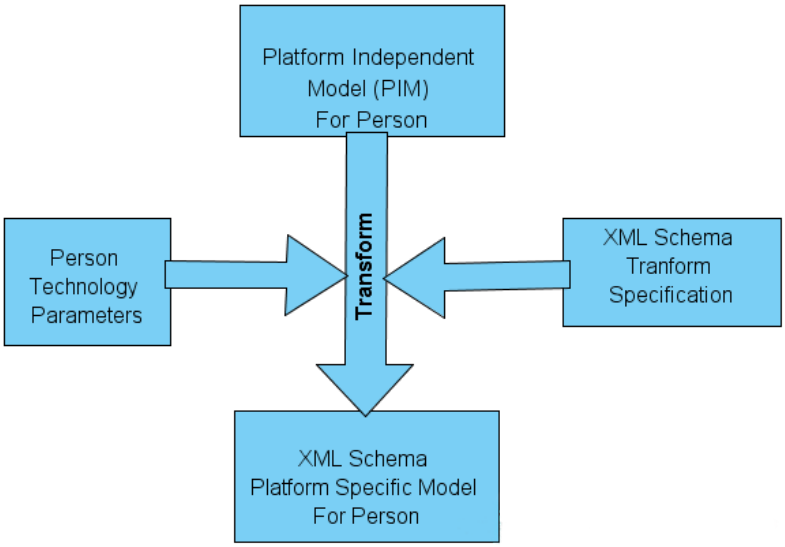
#### 1.1.8.3 Platformně specifický a nezávislý model

V úvodu jsem definoval pojem „platforma“, přičemž pro spuštění systému vyžadujeme vždy nějakou platformu. U implementace transformace či interpretace modelu existují jisté platformní závislosti. Pokud máme model, který je přímo definován pro nějakou jednoznačnou platformu, jedná se o platformně specifický model (PSM). Model, který je absolutně nezávislý na platformě nazýváme platformně nezávislý model (PIM). U MDA konceptu předpokládáme, že je většinou generován PSM (je bližší použité technologii) z PIM, který je blízký modelu procesů a požadavkům systému.

Vlastnosti transformace specifikují, jakým způsobem převést PIM do PSM na základě parametrů závislých na cílové použité platformě. Parametry jsou poskytnuty jako jedna ze vstupních proměnných transformační šablony modelu. Stejný PIM může být na základě jiných vstupních parametrů převeden různě, pro podporu různých technologií. Zároveň také některé technologie mohou umožňovat použití stejného systému – například XML schéma, SQL a uživatelské prostředí.

Příklad transformace PIM do PSM je na obrázku 1.1, včetně vstupních parametrů. Vstupní parametr je XML schéma — definuje, jak bude vypadat výstupní PSM, model a technologické parametry — data, která specifikují dané XML schéma odvozené od použité technologie.

Zároveň výstupní spustitelné řešení nemusí být vždy kompletní. V některých případech může být koncept MDA použit pouze pro část realizovaného systému (definici obchodních procesů nebo rozhraní komponent) a implementace ostatních komponent může být provedena pomocí tradičních programovacích jazyků. Jiné MDA řešení může přímo poskytnout celý systém či systémové komponenty.




Obrázek 1.1: Příklad transformace modelu z PIM do PSM.

Příkladem využití konceptu MDA může být třeba mnou použitý český framework Origam. Další zástupci nástrojů využitelných pro tento koncept jsou uvedeni v odkazech[1].

## 1.2 Framework Origam

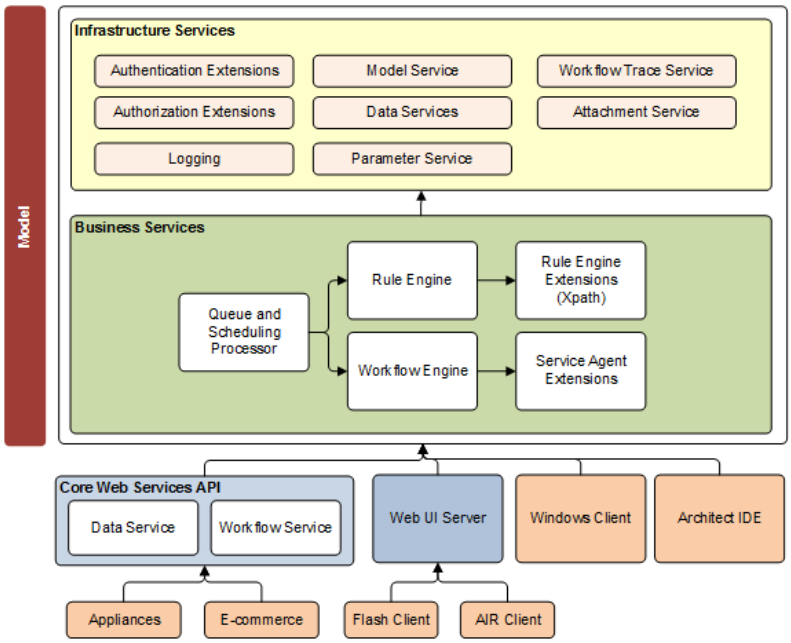
Framework Origam je dílem českých autorů, je stále vyvíjen a vychází nové verze a aktualizace. Jedná se o framework postavený na konceptu vývoje softwaru podle MDA. Počátky vzniku frameworku sahají až do roku 2004, kdy byla vydána první stabilní verze pro produkční prostředí. Díky neustálému vývoji jsou přidávány nové funkcionality a možnosti či zlepšuje stabilitu a podporu dalších platforem a systémů.

Origam je zástupcem konceptu MDA, který využívá k interpretaci modelu (PIM) engine, jenž je specifický pro danou cílovou platformu.

Origam se skládá z následujích částí:

- Vývojové prostředí – Origam Architekt
- Backend Engine – interpretační engine
- Webový a desktopový frontend

V rámci frameworku je součástí také frontendový webový klient, který je zpracován pomocí technologie Adobe Flash®. Framework je multijazyčný a podporuje lokalizace — od samotného uživatelského rozhraní až po lokalizace jednotlivých datových entit. Všechny změny dat jsou zaznamenávány do audit logu aplikace, který se na jednotlivých entitách zapíná či vypíná v modelu.




Obrázek 1.2: Komponenty frameworku Origam

### 1.2.1 Modelování

Pomocí Origam architektu je vytvářen model aplikace, který je výsledně uložen v relační databázi. Z databáze je model interpretován Origam enginem ke spuštění funkční aplikace. Vývoj modelu v Origam architektu probíhá pomocí vlastního jazyka a definic, především pomocí transformací XSLT a jazyka XPath. Vzniklý výsledný model můžeme formálně rozdělit do následujících částí:

1. **Datový model** – datové entity a data struktury, konstanty. Entity slouží primárně k perzistentnímu uložení dat.

2. **Uživatelské rozhraní** – obrazovky, menu, grafy, grafické styly, obrázky (ikony).

3. **Business model** – sekvenční workflow, stavové diagramy, transformace, validační pravidla, fronty zpráv (automatizace úkonů).

4. **Integrační rozhraní (API)** – programovatelné rozhraní pro integraci s dalšími systémy v rámci provozované implementace.

Schéma jednotlivých komponent frameworku Origam a jejich vztahy jsou na obrázku 1.2

### 1.2.2 Komunikační rozhraní pro další systémy

Framework již přímo obsahuje následující integrační prvky pro komunikaci s dalšími systémy, které stačí pouze modelovat:

- WebService API – programovatelné
- REST API – programovatelné
- Volání webových metod
- Import souborů z aplikace MS Excel
- Export souborů do formátu MS Excel (aktuálně zobrazené a filtrované data)
- Uživatelské exporty do MS Excel
- Import CSV a textových dokumentů
- Generování PDF souborů
- Odesílání elektronické pošty přes protokol SMTP
- Příjem emailů ze schránek pomocí POP3 či IMAP a následné automatické zpracování zprávy
- RRS kanál, iCAL kalendáře

### 1.2.3 Provoz a bezpečnost

Vhledem k faktu, že framework ukládá veškerá svá provozní data do relační databáze, je vždy zajištěna kompletní konzistence dat, protože veškeré operace nad daty probíhají v databázové transakci. Pokud by transakce selhala, jsou všechny provedené úpravy vráceny zpět. Veškeré výpočty s daty a validace jsou prováděny přímo na serveru a klientovi je zasílán pouze výsledek — eliminace útoků na straně klienta a možné podvržení dat.

Jednotlivým uživatelům jsou v rámci aplikace přidělovány aplikační role. Podle nastavených rolí jsou filtrována výsledná data, která se odesílají klientovi. O zajištění návratu korektních dat se stará datová vrstva, která neumožňuje vrátit data, na která uživatel nemá oprávnění. Validace oprávnění probíhá na serveru.

### 1.2.4 Technologický přehled

Níže uvádím přehled použitých technologií frameworku v jednotlivých částech:

- Backend – .NET 4.0 Engine

- Server – Windows Server
- Aplikační Server – IIS server
- Reporting a BI – SAP Crystal Reports, SQL Reporting Services nebo další
- UI – Adobe Flash Platform®
- Vývojářské jazyky – MDA, XSLT
- API – Web services, REST
- Úložiště dat – jakákoliv relační databáze, s výhodou MS SQL
- Autentizace – integrovaná databázová, pomocí MS Active Directory či LDAP serveru. Je podporována také vícefaktorová autentizace.

Kapitola 2

# Analýza

Následující kapitola informuje o způsobu, jakým byla zpracovávána analýza navrhovaného informačního systému. Popisuje blíže materiály, které sloužily jako vstup pro analýzu, výstupní dokumenty analýzy, které sloužily pro návrh architektury, a samotný návrh jednotlivých modulů informačního systému.

## 2.1 Vstupní dokumenty analýzy

Hlavním zdrojem specifikace a požadavků na navrhovaný informační systém byl dokument „Specifikace požadavků na vytvoření informačního systému“. Dokument byl dodán jako primární podklad pro vytvoření návrhu dodávaného informačního systému. Jak je z názvu patrné, dokument obsahuje především požadavky na jednotlivé moduly. Tyto požadavky byly často velmi vágně definovány, specifikace občas nedávala smysl či si odporovala v jiné části.

Abychom nedostatky z primárního zdroje doplnili, byly přidány jako podklady pro analýzu ještě následující dokumenty a informace:

- Současný stav informačních systémů společnosti (dokument)
- Současné business procesy zadavatele, jichž se změna týká
- Reálné ukázky funkčnosti konkrétních systémů
- Specifikační schůzky s vedoucím projektu na straně zadavatele (jednalo se o zaměstnance, který se přímo podílel na specifikaci nového systému a věděl, co od něj očekávají)

Nejpřínosnější a nejpřesnější zdroj informací se ukázal projektový vedoucí zadavatele. Dokázal nejpřesněji definovat samotné požadavky na nový informační systém, ale také byl schopen velmi podrobně popsat business procesy v novém informačním systému. Velmi dobře definoval, jaké očekávají chování systému a zpracování jednotlivých požadavků. Zároveň jsme s ním byli schopni

velmi dobře specifikovat stavy některých složitějších položek a přechody mezi těmito stavy v systému.

Popis business procesů byl také důležitý z pohledu, že poskytují samotnému informačnímu systému určitou inteligenci a primární dokument specifikace o nich nemluví vůbec nebo velmi málo. Princip fungování jednotlivých komponent bylo možné odhadnout podle specifikace jednotlivých požadavků, ale nebyla zde záruka stejného vnímání procesu jako na straně zadavatele.

## 2.2 Výstupní dokumenty analýzy

Výstupní dokumenty z analýzy projektu sloužily především pro detailní návrh systému před samotnou realizací prototypu informačního systému.

Jako výstup byla zpracována následující sada dokumentů:

- Seznam případů užití s jejich základním popisem a poznámkou, na jakou část systému mají dopad. Dokument je v příloze D této práce.
- Velmi zjednodušený a základní doménový model, kde jsou zachyceny základní vazby mezi objekty. Výstup posloužil především pro detailní návrh datové vrstvy jednotlivých modulů informačního systému.
- Seznam stavových diagramů – schémata stavů a přechodů mezi nimi u důležitých objektů (entit) v systému.

Během analýzy byla v rámci procesu získávání informací upravena specifikace některých částí informačního systému. Jednalo se především o změny, které pouze drobně měnily proces či jej upravovaly v kontextu větší efektivity a ergonomie koncových uživatelů (interních, externích či obou).

Rovněž bylo, v modulu produktů, upraveno pojmenování některých položek, které nebylo úplně jednoznačné, místy zavádějící až matoucí. V rámci úprav byly položky jednoznačně definovány a specifikovány pomocí hierarchie produktů včetně konkrétních příkladů jednotlivých produktů.

Kapitola 3

# Architektura informačního systému

Tato kapitola se zabývá bližší specifikací navrhovaného systému a jeho rozdělením na jednotlivé moduly, které jsou mezi sebou vzájemně propojeny. Dále se také zabývá výslednou architekturou navrhovaného systému, včetně vrstvy middleware obsahující jednotlivé komunikační rozhraní pro integraci s dalšími firemními systémy.

## 3.1 Bloky informačního systému

Vzhledem ke komplexnosti a rozsáhlosti navrhovaného informačního systému bylo provedeno rozdělení na jednotlivé bloky, které jsou ohraničeny logickým zaměřením své funkcionality. Tyto bloky jsou škálovány na jednotlivé moduly (v rámci frameworku Origam nazývané balíčky). Jednotlivé balíčky rozšiřují základní funkcionality jádra frameworku Origam. Rozdělení informačního systému na moduly a komponenty je pro daný případ následující:

- Blok finančních operací
  - Platební modul
  - Fakturační modul
- Blok marketingových nástrojů
  - Modul produktů
  - Modul ceníku a slev
- Blok komunikace
  - Modul komunikace se zákazníky
  - Modul hromadné komunikace

- Blok správy zákazníků
  - Modul evidence a správy zákazníků
  - Modul evidence a správy uživatelů
- Blok objednávek
  - Objednávkový modul
  - Modul přiřazení produktů (služeb) zákazníkům

Jednotlivé bloky jsou navrženy samostatně a v navrhovaném informačním systému jsou propojeny v balíčku, který spojuje všechny moduly dohromady. V seznamu jednotlivých bloků není uvedena integrační vrstva middleware. Pro každý modul, který potřebuje komunikovat s externím systémem či poskytuje komunikační rozhraní API, je provedeno rozšíření o další balíček s daným komunikačním rozhraním. Tím je zajištěna přehlednost jednotlivých modulů, snadná udržovatelnost a rozšiřitelnost pro další požadované funkcionality.

## 3.2 Architektura informačního systému

### 3.2.1 Architektura

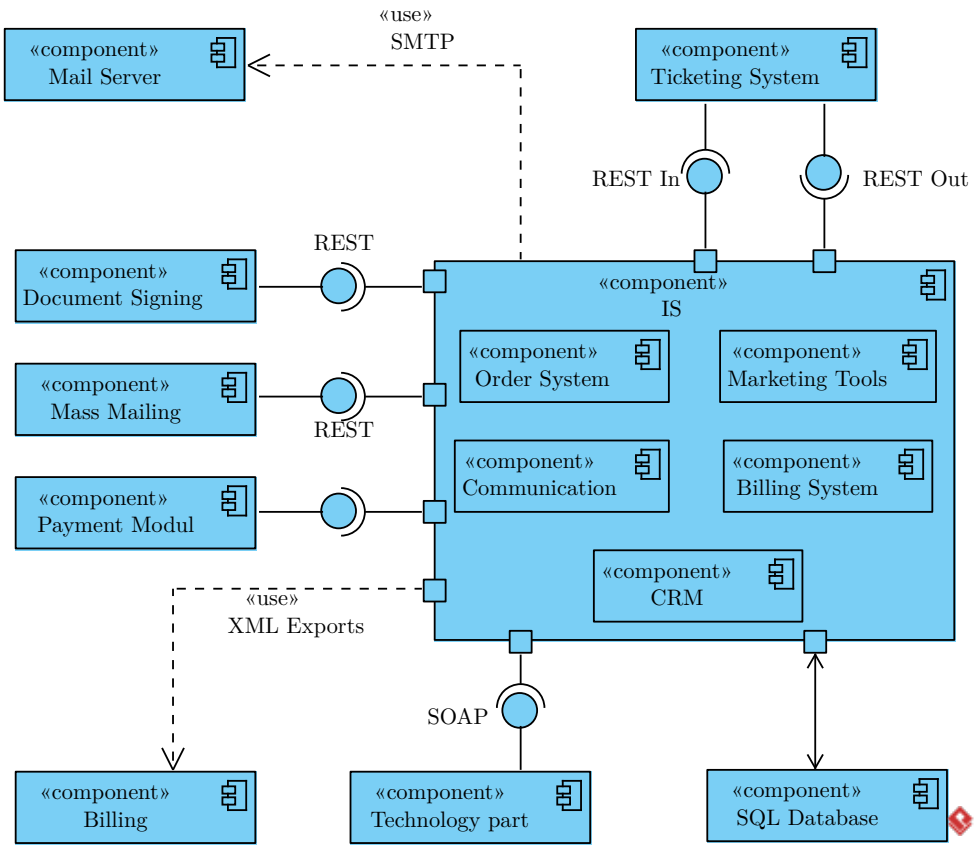
Z pohledu architektury je systém tvořen modelem, který je poté interpretován ve spustitelnou aplikaci pomocí enginu Origam. Samotný model aplikace je interně rozdělený na jednotlivé balíčky obsahující vlastní část modelu, které jsou složeny dohromady ve výslednou aplikaci na nejvyšší úrovni. Z vnějšího pohledu je celý systém tvořen monolitickou aplikací a koncový uživatel nemá jakékoliv informace o interním rozdělení systému na jednotlivé komponenty (submodely).

Informační systém vystavuje okolnímu světu (komponentám, které s ním chtějí komunikovat) komunikační API. V rámci frameworku Origam je část těchto komunikační API přímo integrována – komunikace s SQL, back-endem a webovým klientem. V rámci vývoje jsou již dostupná rozhraní rozšířena o komunikaci s dalšími interními či externími systémy společnosti.

Schéma výsledné architektury včetně všech zamýšlených rozšiřujících rozhraní i s popisem druhu komunikace je zobrazeno na obrázku 3.1.

### 3.2.2 Schéma nasazení

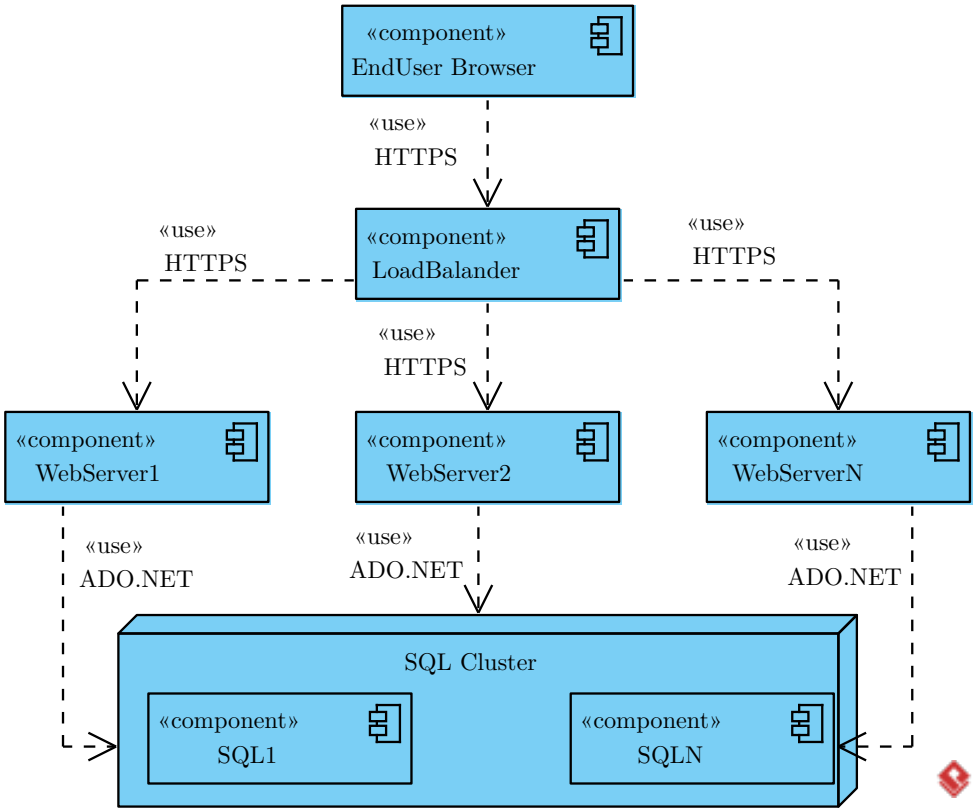
Protože systém bude použit jak pro backoffice, tak pro zpracování klientských požadavků z webové aplikace (webový frontend bude napojen na informační systém přes API – návrh této části není součástí práce), lze předpokládat ve špičkách zvýšenou zátěž a nadměrné množství přicházejících požadavků. Pro vyvážení zátěže je možné celý systém spustit ve více instancích na více




Obrázek 3.1: Architektura informačního systému

serverech. Před tyto instance je nutné jako vstupní prvek umístit load balancer, který rozděluje vstupní požadavky podle zátěže jednotlivých serverů a zároveň se stará o konfiguraci jednotlivých session uživatelů. U load balanceru je další nutnou podmínkou zajištění přesměrování již přihlášeného uživatele na daný server podle cookies, která je vytvořena při přihlášení uživatele do systému či přístupu na webový frontend.

Při vytvoření více instancí je vhodné odpovídajícím způsobem škálovat rovněž databázový server. Jedno z možných schémat nasazení cílového systému včetně využití load balenceru a více instancí informačního systému je na obrázku 3.2.




Obrázek 3.2: Vysoko-zátěžové schéma

Kapitola 4

# Návrh

V rámci návrhu jednotlivých modulů je vždy uveden přehled požadavků a případů užití pro daný modul informačního systému. Nebudu zde uvádět kompletní scénář jednotlivých případů užití. Vzhledem k obsáhlosti a komplexnosti systému to není ani v rámci této práce možné, ale pokusím se jej podat ve zkrácené formě. V analýze a návrhu jednotlivých modulů je zpracován především doménový a stavový diagram k zachycení vazeb mezi jednotlivými objekty a jejich stavy měnící se v čase. Detailní parametry jednotlivých entit nejsou v případě implementace prototypu pro nás důležité, omezím se tedy na nejdůležitější parametry vycházející z podstaty jednotlivých entit. Detailní části jednotlivých entit budou později doplněny v dalších iteracích podle požadavků zákazníka.

Protože se předpokládá implementace systému ve více krocích a jeho postupné rozšiřování, je důležité při návrhu dbát na jednoduchou rozšiřitelnost a udržovatelnost celého modelu aplikace.

Jednotlivé požadavky jsou v zadávací dokumentaci rozděleny pomocí metody prioritizace požadavků „MoSCoW“. V rámci návrhu jsou řešeny především požadavky sledujících typů:

- M – „Must have“
- S – „Should have“
- C – „Could have“, tyto požadavky nejsou řešeny všechny. Pouze ty, které logicky zapadají do navrhovaného modulu a nelze bez nich návrh komponenty provést.

Dále existují ještě požadavky typu W („Would Have“), které jsou řešeny v návrhu velmi minimálně. Zároveň je s nimi počítáno do budoucna a při návrhu je na ně brán zřetel. Seznam funkčních požadavků je uveden v příloze C.

## 4.1 Blok marketingových nástrojů

Hlavní funkcionalitou tohoto bloku informačního systému je evidence a konfigurace služeb, produktů, produktových řad, balíčků a doplňkových služeb. Kromě toho modul obsahuje marketingové nástroje metody up-sellingu a cross-sellingu pro zvýšení prodeje. Poslední částí bloku jsou ceníky, jejich provázanost a samotné ocenění jednotlivých položek.

### 4.1.1 Modul produktů

Jedná se o jeden z hlavních modulů systému. Jeho funkcionalit budou využívat především pracovníci marketingu a zároveň bude také poskytovat služby webovému frontendu (zobrazení produktů, atd.). Tento modul komunikuje hlavně s technologickou částí, která se stará o technické zpracování a vytvoření dané služby pro zákazníka. Zároveň spolupracuje s obchodní částí systému pro případ, že by došlo k nějakému problému, který systém neumí vyřešit sám, a je vyžadován manuální zásah. Komunikace vůči technologické části je blíže probrána v komunikačním rozhraní modulu.

V rámci modulu (dále i celého systému) pracujeme s následující položkami:

1. **Služba** – základní elementární marketingová jednotka. Slouží jako základní stavební blok pro další části.

2. **Produkt** – samostatně objednatelná a ocenitelná položka, skládá se ze služeb.

3. **Produktová řada** – sada produktů obdobného charakteru.

4. **Typ produktu** – třída produktů, které se liší pouze svými parametry, technologií či doplňkovými službami. Jsou vždy z jedné produktové řady.

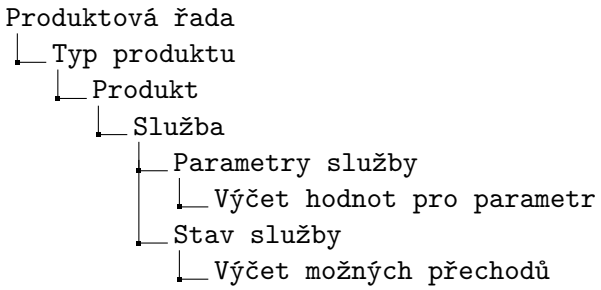
5. **Balíček** – speciální sestavení produktů a doplňkových služeb. Liší se oproti produktům především svojí cenou, konfigurací a doplňkovými službami.

Případy užití v rámci tohoto modulu jsou následující:

- Vytvoření služby v systému – případ užití, který reprezentuje založení, zrušení (deaktivaci) a editaci služby včetně všech navázaných konfigurací.

- Vytvoření závislostí mezi službami – prerekvizity, specifikace závislostí na dalších službách při zřizování v technologiích.

- Vytvoření produktové řady.




Obrázek 4.1: Hierarchie vazeb objektů

- Vytvoření produktu – zde se jedná pouze o vytvoření, editaci a smazání (deaktivaci) produktu (ke smazání přímo docházet nebude z důvodů historizace produktů a také kvůli tomu, že produkty mohou stále běžet u některých zákazníků), tzn. složení produktu z jednotlivých služeb, definice doplňkových služeb a vlastností služeb v produktu (doplňková, samostatná).

- Vytvoření balíčku – vytvoření, editace a smazání (deaktivace) balíčku. Případ užití navazuje na „Vytvoření produktu“.

- Přiřazení balíčku/produktu konkrétnímu zákazníkovi – v produktu/balíčku bude možné editovat, zda má být produkt nabízený všem zákazníkům, pouze vybrané skupině či přímo vybraným klientům.

- Doporučení mezi balíčky a produkty - případ pokrývající specifikaci pro marketingové nástroje up-sellingu a cross-sellingu. Bude sloužit především pro webový front-end, kde budou doporučení zobrazována a nabízena klientům.

Ze vstupních požadavků a případů užití nám vzniká relativně složitá struktura jednotlivých vazeb mezi objekty a informacemi, které je nutné evidovat. Tato hierarchická struktura produktů je zobrazena na diagramu 4.1. Zároveň do hierarchie vstupují ještě balíčky, které mohou být složeny z již stávajících produktů a dalších doplňkových služeb, a také doplňkové služby.

V návrhu začneme specifikací základní elementární jednotky, kterou je služba. Ve službě je potřeba k jejímu efektivnímu fungování dodefinovat parametry, které ji jako samotnou specifikují. Parametry služby jsou samostatný objekt, obsahující seznam nakonfigurovaných hodnot, kde je možné zvolit, jaký typ má mít daný parametr. Služba zároveň má ve své definici uloženo zda je samostatná či doplňková nebo je umožněno obojí. Každá služba ve svých vlastnostech definuje, jestli webový front-end čeká na její odpověď z technologické části. Tento příznak se používá hlavně v kombinaci s validací, zda je daná služba těchto parametrů aktuálně dostupná. K dispozici jsou následující typy parametrů:

- Číslo
- Text
- Výběr

U každého parametru je umožněno definovat výchozí hodnotu z předem definovaného seznamu parametrů. Každá hodnota s sebou nese informaci, zda je povinná či vstupní. Vstupním parametrem myslíme položku, která je nutná pro samotné vytvoření služby v technologické částí a posílá se zároveň s požadavkem na vytvoření do technologií.

Vzhledem ke specifikům jednotlivých služeb je jim možné přiřadit určitý typ. Typ definuje dosažitelné stavy a přechody služby, ve kterých se může nacházet v technologické části při zavádění do provozu. Typ služby je realizován jako kompletně uživatelsky editovatelný stavový diagram.

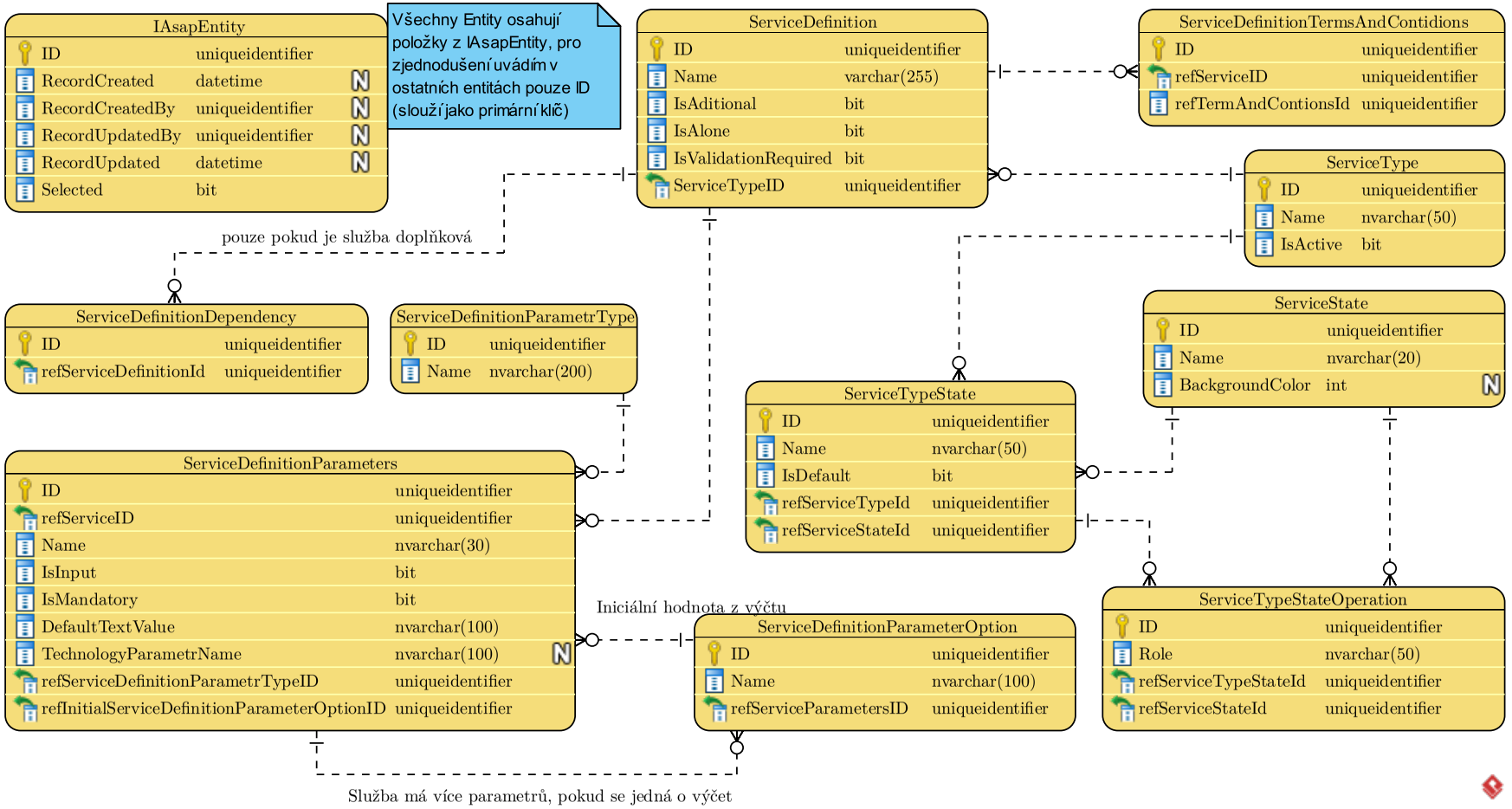
V úvahu rovněž přicházelo využití pevně definovaného stavového diagramu jednotlivých služeb. Tím bychom se ale dostali do velkých problémů, protože všechny služby by měly stejné stavy, což ve skutečnosti takto není. Není možné definovat pro všechny služby jeden průchod stavovým diagramem. Každá služba má různý postup založení – různé stavy v technologii. A zároveň možnost uživatelské editace stavů odbourá zásah do aplikace při zavádění nové služby s rozdílnými stavy. Vše je v rukou uživatelů, kteří mají možnost si typ služby plně nakonfigurovat. Samozřejmě se může stát, že služba bude mít špatně definované stavy – špatné přechody. V tom případě předpokládáme určitou znalost nutnou pro definici jednotlivých stavů, aby vše dávalo smysl.

Podrobné schéma jednotlivých vazeb mezi objekty tvořícími konfiguraci služby a nastavení služby je zachyceno na diagramu 4.2.

Produktovou řadu můžeme specifikovat jako bližší kategorizaci produktu. Jedná se pouze o seznam, který lze uživatelsky definovat. Hodnota produktové řady se vybírá přímo v definici konkrétního produktu či balíčku.

Nyní přejdeme k návrhu objektu produktu, který se skládá z výše definovaných služeb a zároveň patří do produktové řady. Hlavní částí produktu je definice služeb. Předpokladem pro existenci produktu v systému je, aby obsahoval minimálně vazbu na jednu až N elementárních služeb, které jsou definovány jako samostatné. Dále je možné k produktu přiřadit 0 až N služeb, které jsou označeny jako doplňkové. Zároveň je ve výčtu služeb u daného produktu nastavena závislost a návaznost nutná při zřizování v technologické části – jestli službu lze zřídit přímo nebo existuje nějaká závislost a prerekvizita na jiné služby z definovaného produktu.

V definici produktu má uživatel možnost zadat další produkty stejného typu, mezi kterými je možné přecházet – většinou jsou odlišeny jinou specifikací jednotlivých parametrů, buď vyšší či nižší konfigurace. Dále je objekt rozšířen o definici kategorie uživatelů, kterým je balíček určen – výčet uživatelů, či skupin pro něž je produkt speciálně nabízen. Vše je plně konfigurovatelné uživatelem.




Obrázek 4.2: Datový model služeb

Poslední částí je vytvoření balíčku. Balíček je tvořen sestavením 2 a více produktů do kompletu a dále žádnou nebo více doplňkovými službami. V případě balíčku je možné jednotlivým produktům specifikovat vlastnosti, jestli je povinný, nepovinný či zaměnitelný. U zaměnitelného produktu v balíčku je možné vybrat pouze z definovaných položek u variant produktu. Konfiguraci položek balíčku je možné využít pouze za předpokladu, že v balíčku stále zůstanou po odebrání nepovinných částí vždy minimálně 2 produkty. Kontrola správného složení balíčku je finálně softwarově ošetřena. Pokud balíček nesplňuje základní kritéria, nelze jej vůbec uložit do systému. Jinak by vytváření balíčků v systému nemělo cílený význam a stávaly by se z nich klasické produkty. Balíček je vytvořen nad objektem produktu – technicky se jedná pouze o další produkt, který má rozšířené vlastnosti a sám se skládá z jednotlivých produktů. Jednotlivé položky balíčku jsou definovány ve vlastním objektu.

Schéma jednotlivých vazeb mezi objekty v rámci produktů, balíčků a jejich nastavení je zobrazeno na diagramu 4.3.

Poslední částí tohoto modulu je návrh komponenty, která slouží pro definici vazeb pro up-selling a cross-selling mezi jednotlivými produkty. Tento marketingový nástroj najde své uplatnění především ve webovém front-endu aplikace pro koncové uživatele. Slouží k motivaci zákazníka nakoupit další související produkty či služby k objednávanému produktu nebo přímo jeho vyšší konfiguraci.

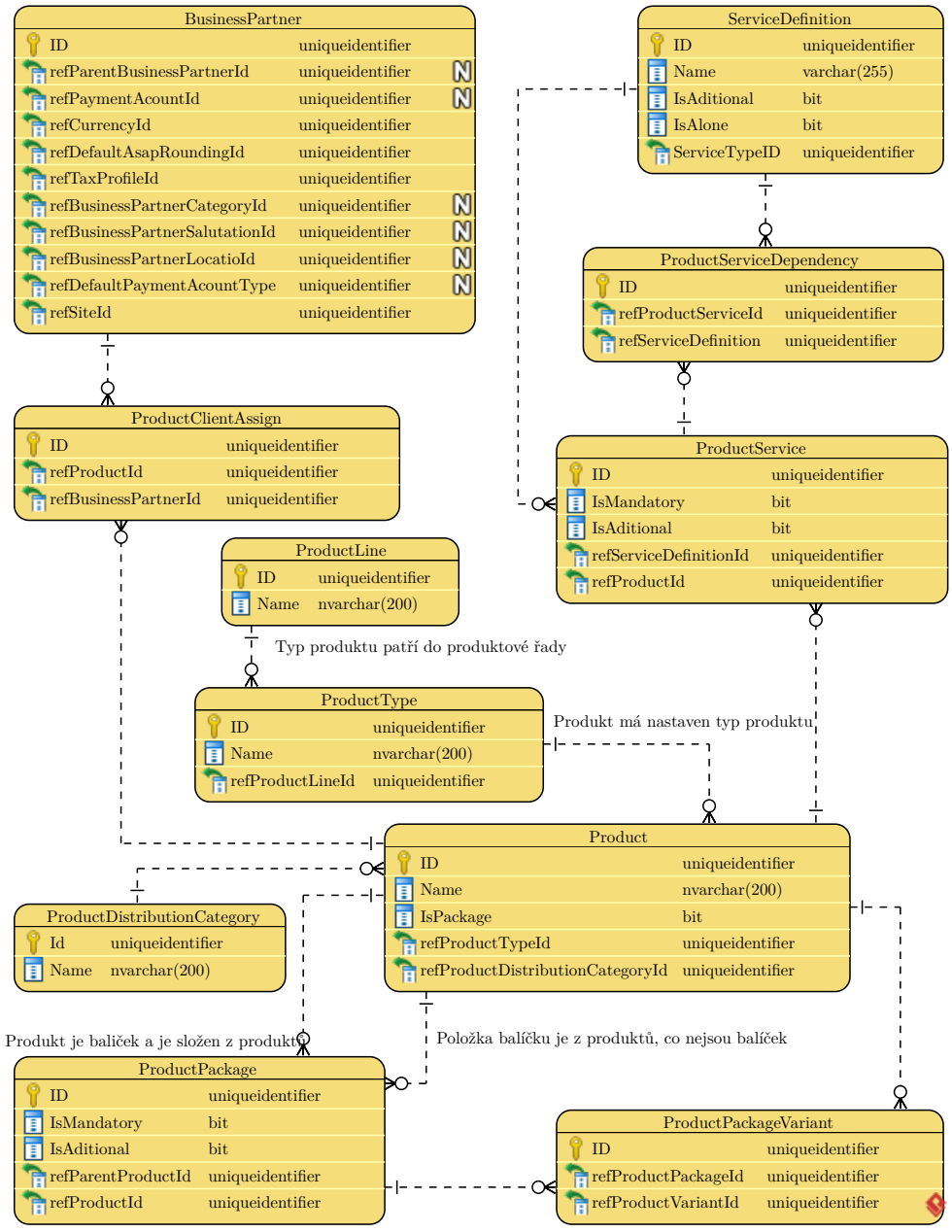
Tato komponenta je řešena pomocí matice prostupnosti – pro každý produkt je definován výčet produktů pro up-selling, cross-selling a také down-selling. V rámci implementace ve frameworku Origam je zpracována jako pole hodnot. Vazby mezi objekty v matici propustnosti jsou zobrazeny na diagramu 4.4.

### 4.1.2 Komunikační rozhraní modulu produktů

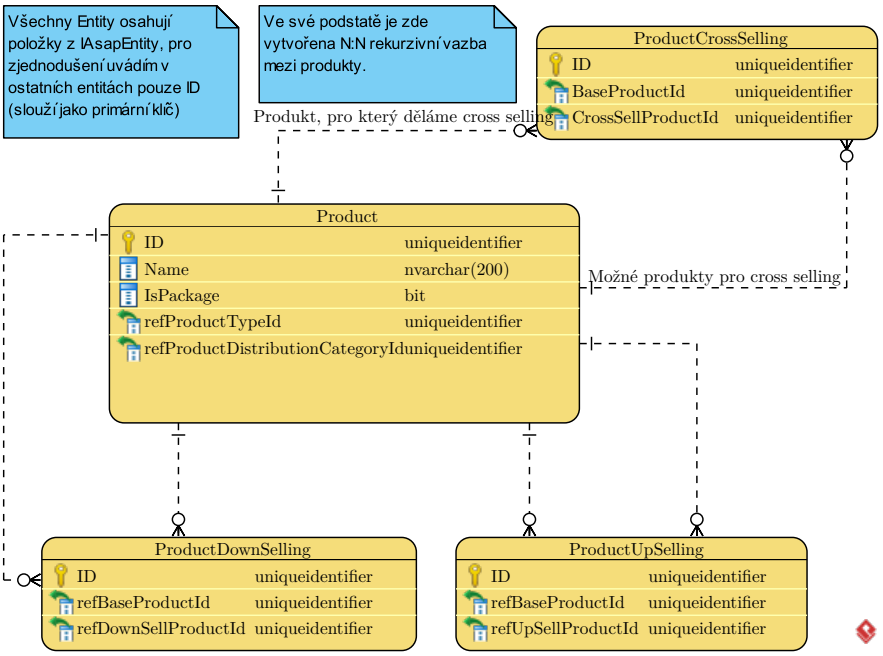
Vzhledem k tomu, že většina definovaných softwarových služeb, které v systému jsou, je technického rázu, je nutné navrhnout vhodnou komunikaci mezi informačním systému a technickou částí. Navrhovaný informační systém se stará pouze o evidenci objednaných a běžících služeb, technická realizace bude probíhat v jiném systému. V návrhu je tento požadavek již zahrnut v definici služby – má definované parametry pro technologickou část. V rámci komunikace mezi systémy potřebujeme rozlišit dva druhy komunikace:

1. **Synchronní** – systém čeká na přímou odpověď od technické části, než bude pokračovat ve své práci. Jedná se například o validaci dostupnosti služby.

2. **Asynchronní** – systém nečeká na přímou odpověď a požadavek je zařazen do fronty zpráv a následně odeslán do technologické části. Technologie vrací po zpracování odpověď na daný požadavek.




Obrázek 4.3: Datový model produktů a balíčků




Obrázek 4.4: Datový model cross a up-sellingu

Protože technologická část systému, se kterou budeme komunikovat, je vyvíjena zároveň s business částí informačního systému, byl domluven komunikační protokol ve formě SOAP/XML zpráv. Zprávy do technologického systému budou zasílány pomocí HTTP requestu, který bude obsahovat veškeré potřebné informace pro vyřízení. Odesílaná zpráva bude mít dva možné formáty dle typu zprávy, kterou odesíláme. Ukázka předpokládaného formátu synchronní zprávy je na obrázku 4.5.

Synchronní zprávy jsou odesílány z business části do technologické části na jeden společný endpoint. Technologická část si poté podle obsahu zprávy (podle identifikátoru typu služby) zpracuje zprávu dále. Technologická část není součástí této práce. Nutným předpokladem pro správnou funkčnost je správná synchronizace seznamu typů služeb a jejich unikátních identifikátorů napříč systémy. Na synchronní zprávy technologická část odpovídá přímo hodnotou v návratové zprávě – aktuálně předpokládáme pouze návratovou hodnotu „true“ či „false“.

Dalším typem zprávy je asynchronní zpráva, která je odesílána v případě, že se jedná o zřízení nové služby, zrušení či změnu služby. Systém tyto zprávy postupně zařazuje do fronty. Z fronty zpráv je systém automaticky odesílá na předem definovaný endpoint technologického systému pro asynchronní zprávy. Příklad asynchronní zprávy je na obrázku 4.6.

Nyní se blíže podívejme na strukturu asynchronní zprávy a její obsah. Zpráva se skládá ze tří základních částí:

```
<service:validate>
  <id>GUID</id>
  <type>Typ sluzby - GUID</type>
  <name>Jmeno sluzby</name>
  <!-- typ operace -> vytvoreni, zruseni, atd -->
  <action>Operace</action>
  <parameters>
    <!-- nazev parametru je z~definice sluzby -->
    <param name="nazev1">hodnota</param>
    <param name="nazev2">hodnota</param>
    <param name="nazev3">hodnota</param>
  </parameters>
</service:validate>
```

Obrázek 4.5: Ukázka vnitřní struktury synchronní zprávy

```
<!-- nazev metody=typ provadene operace -->
<service:new_service>
  <id>GUID</id>
  <type>Typ sluzby - GUID</type>
  <name>Jmeno sluzby</name>
  <!--typ operace,pro evidenci -->
  <action>Operace</action>
  <parameters>
    <!-- nazev parametru je z~definice sluzby -->
    <param name="nazev1">hodnota</param>
    <param name="nazev2">hodnota</param>
    <param name="nazev3">hodnota</param>
  </parameters>
  <states>
    <state name="success">
      <uri>adresa/api/service/success/GUID</uri>
    </state>
    <state name="fail">
      <uri>adresa/api/service/fail/GUID</uri>
    </state>
  </states>
</service:new_service>
```

Obrázek 4.6: Ukázka vnitřní struktury asynchronní zprávy

- **Definice zprávy a základních parametrů** – zde je uvedena metoda, která se v rámci zprávy zavolá. Prováděná operace vychází z metody, protože proces při vytvoření, zrušení či prodloužení služby je dle informací ze zadávací dokumentace různý. Dále jsou v hlavičce zprávy uvedeny základní identifikátory služby v systému.

- **Definice parametrů služby** – ve zprávě jsou definovány parametry služby a jejich hodnoty. Hodnoty jsou získány ze samotné definice služby v objednávce. Je požadováno, aby byly zadány alespoň povinné parametry. Jinak nelze službu založit.

- **Definice stavů** – možné stavy služby, do kterých může přejít. Stavy jsou u zakládané služby měněny pomocí API. Výčet možných stavů je definován v typu služby.

Kromě odesílání zpráv je v modulu implementováno notifikační REST API. Toto rozhraní slouží technologické části pro možnost změnit stav jednotlivých vytvářených služeb. Změny stavu jednotlivých služeb jsou dále navázány na další zpracování v systému, ať už se jedná o další operaci s jinou službou či akci typu fakturace nebo vystavení jiného dokladu. Struktura REST API adresy ve formátu **adresa/api/service/state/GUID** má následující význam:

- **adresa/api** – definuje adresu serveru informačního systému

- **service** – položka definující typ služby

- **state** – název stavu služby, který chceme aktuálně nastavit

- **GUID** – unikátní identifikátor objednávané služby v systému z objednávky

Všechny možné adresy (stavy) služby jsou přímo zasílány v rámci zprávy. Přímým přistoupením na adresu rozhraní a zadáním jednoznačného identifikátoru proběhne spuštění procesu, který změní současný stav služby na stav, který voláme přes API. Samozřejmě v rámci této změny proběhne další validace, zda je vůbec možné daný stavu u služby nastavit. Na změnu stavu služby je možno v rámci systému navázat další akci.

### 4.1.3 Modul ceníků a slev

Hlavní náplní funkcionality tohoto modulu je uchovávat informace o cenách jednotlivých produktů a zároveň také poskytovat nástroje na operace s ceníky. Dále je možné jednotlivé ceníky přiřazovat fakturačním kontaktům alias uživatelům. Další nástroj modulu je definice slev. Mohou existovat dva hlavní typy ceníků:

1. **Hlavní ceník** – hlavní ceník slouží vždy jako primární informace o ceně. Ceník obsahuje všechny produkty a balíčky v nabídce zadavatele včetně všech operací, které s danou položkou mohou být prováděny. Každý klient (fakturační kontakt) musí mít vždy přiřazen hlavní ceník. Hlavní ceník může být přiřazen pouze jeden.

2. **Vedlejší ceník** – vedlejší ceník přidává k hlavnímu ceníku různé zvýhodnění, akce, slevy na definované produkty. Nemusí zde být definované všechny produkty, což jej odlišuje od hlavního ceníku. Vedlejší ceník může být přiřazován automaticky na základě různých kritérií.

Modul svojí funkčností pokrývá tyto případy užití:

- **Vytvoření a správa hlavního ceníku** – případ užití zahrnuje veškeré operace ohledně hlavního ceníku od jeho vytvoření až po správu.
- **Vytvoření a správa vedlejších ceníku** – rozšiřující ceníky.
- **Ocenění** – pod tímto případem užití rozumíme přiřazení ceny balíčku či produktu k dané službě a operaci. Znamená to, že pro konkrétní situaci systém poskytne jednoznačnou cenu podle zákazníka, který produkt objednává.
- **Vytvoření a správa ceníku oprávněných nákladů** – znamená vytvoření speciálního ceníku, který slouží k ocenění produktu/balíčku při jeho zrušení. Neplatí, že se vždy vrací celá částka klientovi. Část nákladů je nevratných.
- **Vytvoření a správa interního ceníku nákladů** – případ užití pokrývající kompletní management speciálního ceníku pro určování interních nákladů jednotlivých operací v rámci společnosti, které jsou prováděny. Tento požadavek je především pro následné interní vyhodnocení ekonomických hledisek. Z pohledu užití systému nemá žádnou reálnou hodnotu.
- **Použití množstevní slevy** – v případě objednání více položek stejného druhu umožní ceník aplikaci množstevní slevy.

Vedlejší ceníky je možné přiřazovat buď přímo staticky jednotlivým klientům nebo je může automaticky přiřazovat systém dle různých kritérií. Vedlejší ceníky zároveň slouží jako aparát pro aplikaci slev pro jednotlivé produkty.

Protože hlavní a vedlejší ceník obsahují stejné základní informace, je k jejich definici použit společný objekt. Společné položky v cenících jsou následující:

- Název ceníku
- Druh ceníku – prodejní či nákupní

- Stav
- Obrat do – speciální faktor pro dynamické přiřazení ceníku
- Hodnota objednaných služeb – speciální faktory pro dynamické přiřazení ceníku

Pro konkrétní zadání jednotlivé ceny produktu slouží cenový řádek. Každý cenový řádek se váže pouze k jednomu definovaného ceníku. Abychom byli v rámci cenového řádku jednoznačně schopní identifikovat k čemu daná cena patří, je toto rozlišení určeno pomocí následujících položek v cenovém řádku:

- Balíček, resp. produkt
- Produkt (pokud se nejedná o balíček, je zde zobrazen původní produkt)
- Služba – služba z daného produktu/balíčku
- Operace

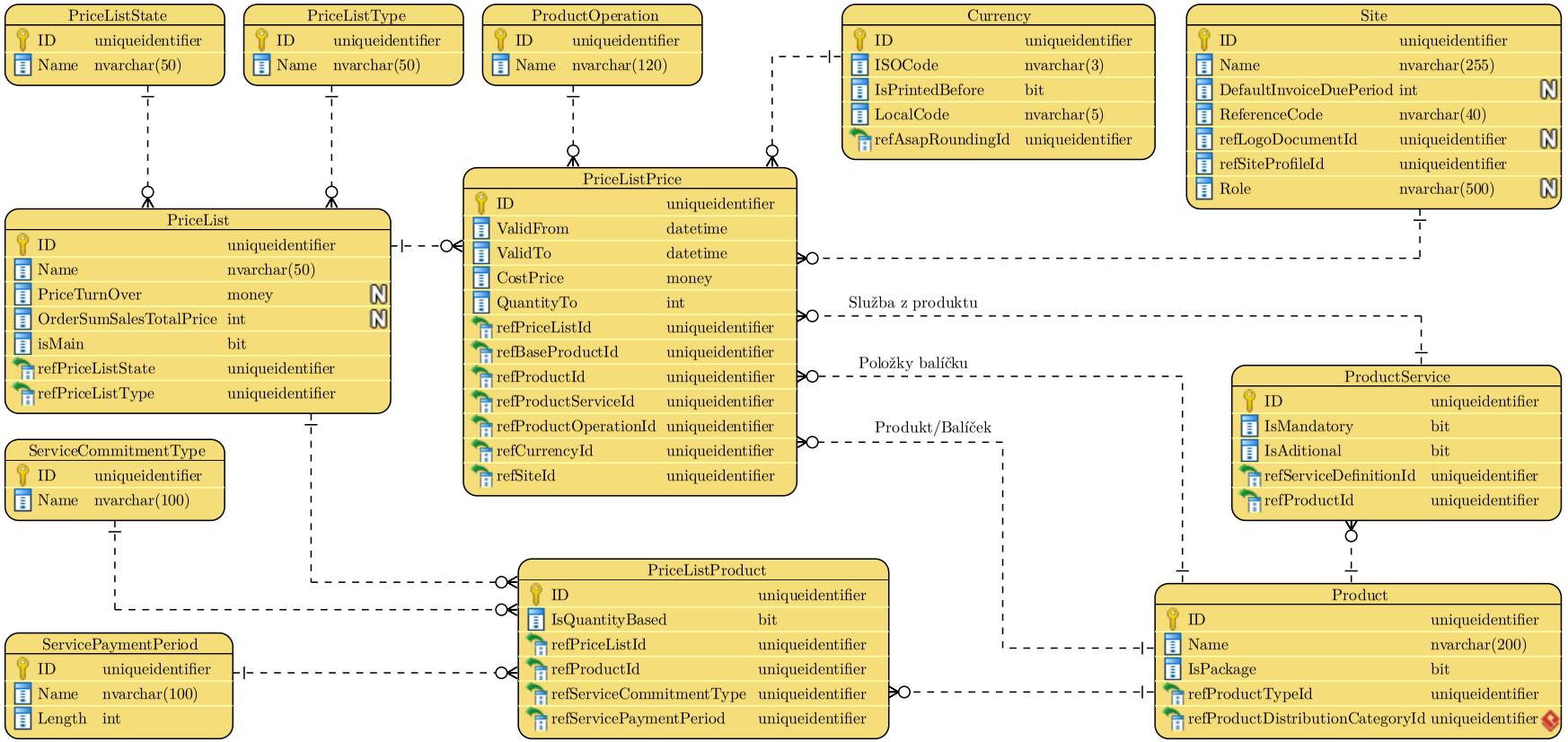
Dále je v každém cenovém řádku uvedena další informace o konkrétní ceně a jejich vlastnostech:

- Platnost od do
- Cena
- Měna
- Zda existuje stupňovaná cena – ekvivalent množstevní slevy, daná cena platí pro konkrétního množství
- Do množství – hodnota pro stupňovanou cenu
- Délka platebního období – doba, po kterou bude služba objednána
- Délka závazku
- Nákladová cena (pouze pro hlavní ceník)
- Cena oprávněného nákladu (pouze pro hlavní ceník)
- Zkušební období (doba, po kterou je možné produkt bezplatně zrušit)

Definicí cenového řádku jsme splnili i požadavky na vytvoření ceníků oprávněných nákladů a ceníku interních nákladů. Tyto položky jsou přímo zadány u jednotlivých produktů v položce cenového řádku konkrétního hlavního ceníku.

Vazby mezi jednotlivými objekty v rámci ceníku a cenového řádku jsou na diagramu 4.7.

Podívejme se tedy nakonec na způsob získání ocenění jednotlivé položky, kterou uživatel objednává. Postup je následující:




Obrázek 4.7: Datový model ceníků

1. Vezmou se ceníkové položky a profiltrují se podle ceníků, které má daný uživatel nastaven. Minimálně má vždy jeden ceník nastaven – hlavní ceník.

2. Ceníkové položky se dále filtrují podle daného produktu (balíčku), služby a prováděné operace.

3. Výsledkem je list ceníkových položek, který se seřadí podle nejvýhodnější ceny.

4. Nejvýhodnější cenou je položka zákazníkovi oceněna.

Tímto postupem zajistíme nejvýhodnější cenu a zároveň nám zůstává velmi transparentní struktura a prostupnost jednotlivých ceníků mezi sebou. Rovněž jsme splnili definici cenového řádku a možnost zadat stupňovanou cenu, která odráží požadavek na množstevní slevu.

Dále je celý slevový aparát akčních cen řešen formou doplňkových ceníků, čímž máme splněný požadavek na zadání a využití slev.

Tento modul nebude obsahovat žádné externí komunikační rozhraní (rozhraní pro GUI neuvažujeme, není součástí návrhu). Součástí modulu je možnost importu ceníků pomocí speciálního procesu. Protože ceník obsahuje vazby na další objekty v systému, je nutné při importu ceníku vazby nastavit korektně, jinak bychom ceník dostali do nekonzistentního stavu. Proces importu ceníků do systémů je podporovaný v následujícím formátu:

- Vstupní formát dat bude sešit formátu Microsoft Excel či ve formátu CSV.

- Formát sešitu musí obsahovat minimálně položky vypsané níže. Možné formáty položek jsou buď textové či jedinečné identifikátory, které jsou v databázi u daných položek.

  - Balíček
  - Produkt
  - Služba
  - Operace
  - Cena
  - Datum od do
  - Měna
  - Platební období

- Import ceníku probíhá v sekvenčním workflow a přidávají se ceny pouze pro jeden vybraný ceník.

## 4.2 Blok objednávkového systému

Hlavní funkční náplní tohoto bloku informačního systému a jeho modulů je zpracování klientských požadavků na operace se službami, jejich evidence a zpracování dalších požadavků týkajících se klientských účtů a jejich služeb. Tento blok systému se oproti předchozímu liší především ve svém zacílení – původní modul byl spíše technického a konfiguračního rázu, tento je více orientovaný směrem na klienty, jejich požadavky a bude poskytovat v budoucnu většinu informací pro webový front-end.

Tento blok má dva hlavní moduly:

- Objednávkový modul
- Modul přiřazení produktů (služeb) zákazníkům

### 4.2.1 Objednávkový modul

Modul je klíčový pro klienty společnosti. Přes něj mohou provádět veškeré operace se svými zakoupenými službami či nakupovat a rozšiřovat stávající. Modul pokrývá následující případy užití z pohledu uživatele:

- Přiřazení doplňkového ceníku dle závislých faktorů
- Zobrazení dostupných produktů/balíčků uživateli k objednání
- Možnost změny aktuálně objednaných produktů uživatele
- Prodloužení produktu/balíčku uživatelem
- Ukončení produktu/balíčku uživatelem
- Objednání speciální služby – služby pro správu (speciální kategorie služeb)
- Přidání položek do košíku
- Odeslání objednávky uživatelem ke zpracování
- Udělení souhlasu s VOP (Všeobecné obchodní podmínky)

Dále je nutné, aby systém uměl zpracovat v procesu objednávek následující požadavky:

- Uložení objednávky při odhlášení uživatele
- Nahrání uložené objednávky a validace aktuálních cen
- Uložení anonymní objednávky nepřihlášeného uživatele

- Provedení platby – tímto myslíme, že pokud není objednávka za nulovou cenu, je uživatel vyzván k zaplacení své objednávky. Poté probíhá další zpracování objednávky.
- Kontrola zaplacení objednávky
- Notifikace nezaplacených objednávek
- Zrušení nezaplacených objednávek po termínu expirace

Z případů výše vidíme, že tento modul nemá příliš velké množství evidenčních požadavků mimo objektu objednávky, ale spíše se jedná o zpracování business procesů jednotlivých úkolů pro objednávky. Podívejme se blíže na datovou strukturu modulu a jím spravovaná data. Modul frameworku Origam sice disponuje objednávkovým systémem, jedná se ale pouze o objednávky klasických produktů, které jsou navázány na sklady, dodací listy a expedici. My potřebujeme evidovat mnohem složitější a komplexnější datové struktury.

Hlavní datovou částí tohoto modulu je objekt objednávky. Objekt objednávky je přiřazen přímo uživateli (fakturačnímu kontaktu), který objednávku provedl. Dále evidujeme složení celé objednávky a informace pro její samotné zpracování. Vyžadujeme minimálně následující hodnoty:

- Uživatel – uživatel, který objednávku vytvořil. Pokud se bude jednat o anonymní objednávku, bude zde vyplněno „anonymní“.
- Datum objednání
- Stav – na stav objednávky jsou navázány další procesy pro zpracování, stav objednávky je odvozen podle stavu vyřizování jednotlivých objednaných produktů a jejich stavech technologické části a případného úspěchu či neúspěchu.
- Cena – celková cena objednávky
- Fakturační údaje – nutné pro zpracování v účetnictví, platbu a fakturaci
- Způsob platby – je velmi úzce provázán se stavem objednávky. Různé způsoby platby mají rozdílné zpracování objednávky.

Základní objekt objednávky máme definovaný, nyní k němu potřebujeme přidat jednotlivé položky, které si ve skutečnosti uživatel objednal, a jejich parametry. Na objednávky je navázán další objekt, který uchovává jednotlivé řádky objednávky, neboli „produkty zákazníka“. Každý řádek objednávky obsahuje následující evidenční informace:

- Balíček/Produkt
- Produkt

- Služba – konkrétní služba z balíčku či produktu
- Cena
- Délka platebního obdobní
- Stav – primárně zřízená/nezřízená/ukončená a další stavy
- Uživatelský popis – pro lepší orientaci zákazníka
- Datum expirace
- Typ závazku
- Detaily služby z produktu a její parametry

Jednotlivé položky objednávky rozšiřují svoji specifikaci dalším objektem „detail řádku“, který eviduje jednotlivé služby z produktu a jejich parametry zadané zákazníkem. Všechny položky vychází z definice služeb v části 4.1.1.

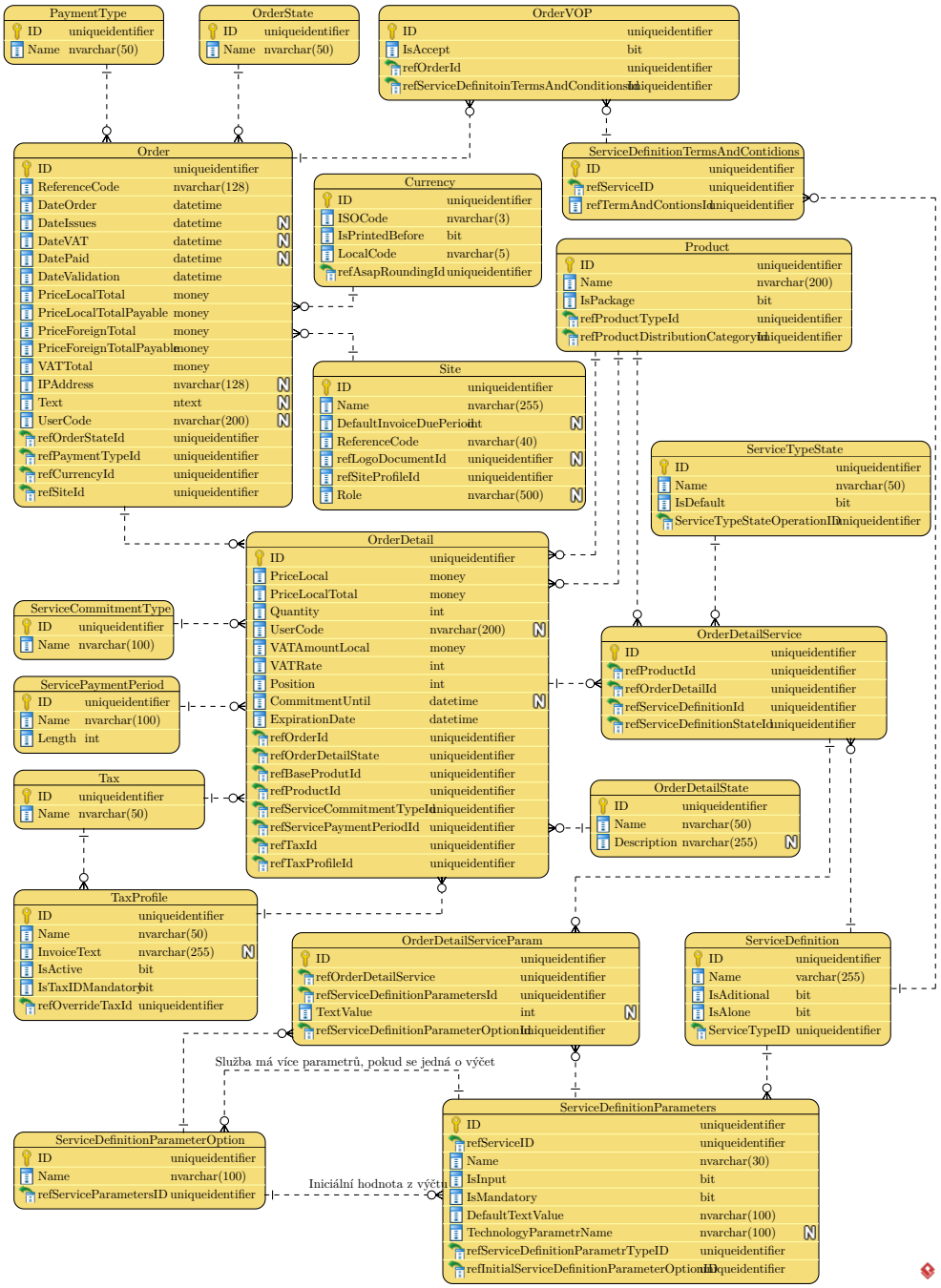
Datové schéma pro objednávky je zobrazeno na diagramu 4.8.

Tímto bychom měli navrženou datovou strukturu, nyní následuje samotné pojednání o realizaci objednávky v systému, bližší specifikace použití objednávky a košíku v rámci kontextu celého informačního systému.

Výše jsem zmínil dva pojmy: objednávka a košík. Je mezi těmito pojmy nějaký faktický rozdíl? Mezi těmito pojmy skutečně žádný rozdíl v kontextu systému není. V případě košíku se jedná také o objednávku, jen ve stavu nerealizováno (případně uloženo nebo nová). Princip fungování objednávek je takto navržený, protože potřebujeme splnit zadané systémové požadavky:

- **Přidání produktu do košíku** – v systému je ihned založena nová objednávka s daným produktem či produkty. Ve chvíli vytvoření se jedná o anonymní objednávku. Uchování anonymních objednávek je jeden z požadavků na navrhovaný systém. Objednávka je v tomto případě označena jako nerealizovaná. Každý produkt v košíku je veden jako jeden řádek objednávky a má také v datovém modelu vyplněny odpovídající detaily.
- **Uživatel je přihlášený a má položky v košíku** – případ stejný jako v předchozím bodu, pouze s rozdílem, že objednávka už nebude vedena jako anonymní, ale přímo se přiřadí k danému uživateli. Je to z důvodů uchování rozpracovaných objednávek mezi přihlášeními do systému, které mohou být později odeslány ke zpracování.

Jakýkoliv pohyb v košíku znamená vytvoření objednávky v systému se všemi požadovanými detaily a zároveň jsou všechny objednávky ukládány do systému. Neřešíme, jestli objednávka byla skutečně objednána či ne.




Obrázek 4.8: Datový model objednávek

V požadavcích máme jednodušší proces, kdy se jedná pouze o zobrazení produktů/balíčků, které má dostupné daný uživatel. Tento proces je řešen pomocí jednoduchého filtru na uživatele vůči celkovému datasetu všech aktivních produktů v systému. Zároveň s produkty systém dodá i jejich aktuální ceny pro uživatele.

Ještě je nutné popsat způsob přiřazení doplňkových ceníků uživateli. Doplňkové ceníky jsou uživateli přiřazovány v různých okamžicích:

1. Periodicky pomocí interního plánovače
2. Po přihlášení uživatele do systému
3. Před odesláním objednávky do systému – mohlo dojít k přiřazení lepšího ceníku podle celkové hodnoty objednávky

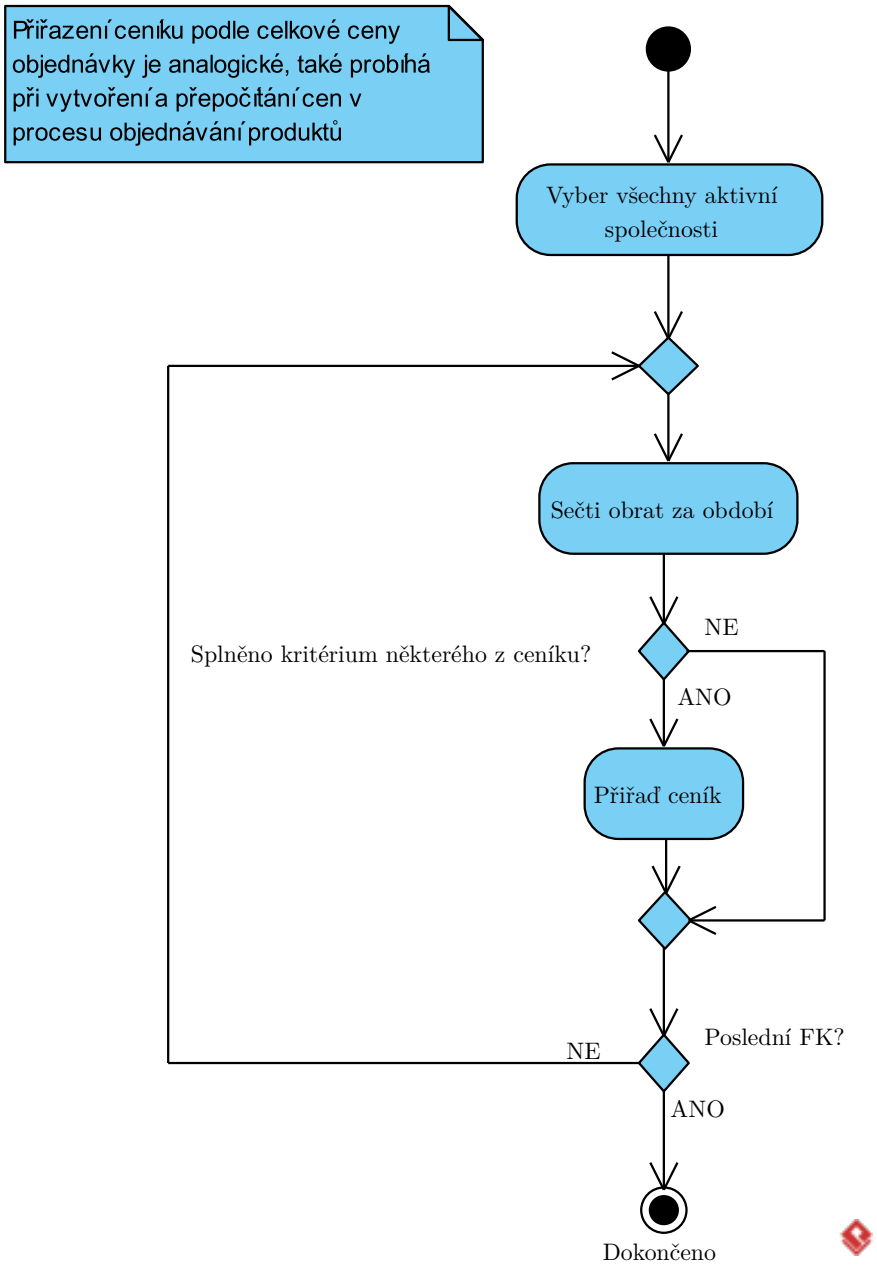
Workflow zajišťující tuto činnost provede analýzu dat zákazníka dle specifikovaných požadavkům pro přidělování ceníků a doplňkové ceníky jsou mu buď přiřazeny či odebrány, pokud se dostane do situace, že daný limit/požadavek již nesplňuje. Proces přiřazení ceníků je zachycen na obrázku 4.9.

Nyní se můžeme přes datový model a obecné fungování objednávek přesunout k návrhu procesů zodpovědných za zpracování objednávky a úkonů s objednávkami. Další procesy můžeme rozdělit do dvou skupin:

1. Operace s produkty – zde se jedná o více různých procesů jako objednání, prodloužení, zrušení produktu.
2. Zpracování objednávky – zpracování objednávky po jejím odeslání do systému.

Pro implementaci zpracovaní objednávky je využito sekvenční workflow, stavové automaty a zpracování pomocí front zpráv ve frameworku Origam. Tento proces pokrývá zpracování objednávky po vložení produktů a jejich specifikací do košíku. V tuto chvíli již uživatel zná celkovou cenu objednávky a ta je připravena po schválení VOP k odeslání ke zpracování. Proces zpracování se skládá z posloupnosti následujících kroků a je zachycen na obrázku 4.10:

- **Souhlas s VOP** – uživatel odsouhlasí VOP před samotným odesláním objednávky. Odsouhlasené podmínky se vážou vždy pouze k danému produktu (také žádné mít nemusí). Odsouhlasení se ukládá v systému u každého uživatele (jeho fakturačního kontaktu). Pokud uživatel již podmínky pro daný produkt odsouhlasil, nebude je odsouhlasovat znovu. Vše se provede pouze v případě, že se bude jednat o nové či novou revizi již existujících podmínek. Uživateli se zobrazí před odesláním objednávky všechny VOP, které nemá pod účtem, k odsouhlasení pomocí jednoduchého zatržítka. Bez odsouhlasení nebude možné v objednání pokračovat a objednávku nebude možné odeslat.




Obrázek 4.9: Přiřazení závislých ceníků

- **Odeslání objednávky** – po odsouhlasení podmínek se objednávce nastaví stav na přijatá a provádí se její zpracování.

- **Zaplacení objednávky** – objednávka už je zařazena v systému. K jejímu dalšímu zpracování je nutná platba (pouze v případě, že objednávka nemá nulovou cenu). Podle typu platby zvolené uživatelem systém realizuje další kroky.

Dále potřebujeme zajistit, že objednané služby z objednávky se po zaplacení automaticky odešlou ke zpracování do technologické části. Toto je zajištěno pomocí kombinace sekvenčního workflow, stavových diagramů a front zpráv. Pokud je provedena platba, je spárována v systému v modulu plateb a zároveň se změní stav objednávky na zaplaceno. Celý stavový diagram objednávky je zobrazen na obrázku 4.11 Ke slovu se tímto dostává workflow služeb a jejich vytvoření v technologické části.

Zároveň na pozadí systému periodicky probíhá kontrola nezaplacených objednávek pomocí interního plánovače, který spouští sekvenční workflow kontrolující expiraci objednávky. Workflow má dvě hlavní části:

1. **Kontrola expirace** – v tomto kroku se ověřuje, zda je objednávka stále platná. Objednávka může v systému existovat pouze definovanou dobu, po uplynutí této lhůty je zrušena.

2. **Notifikace nezaplacených objednávek** – v rámci čekací doby na platbu je uživatel v definovaných sekvencích upozorněn, že má v systému nezaplacenou objednávku a je doporučeno zaplacení.

Proces kontroly zaplacení objednávky včetně notifikace je na obrázku 4.12.

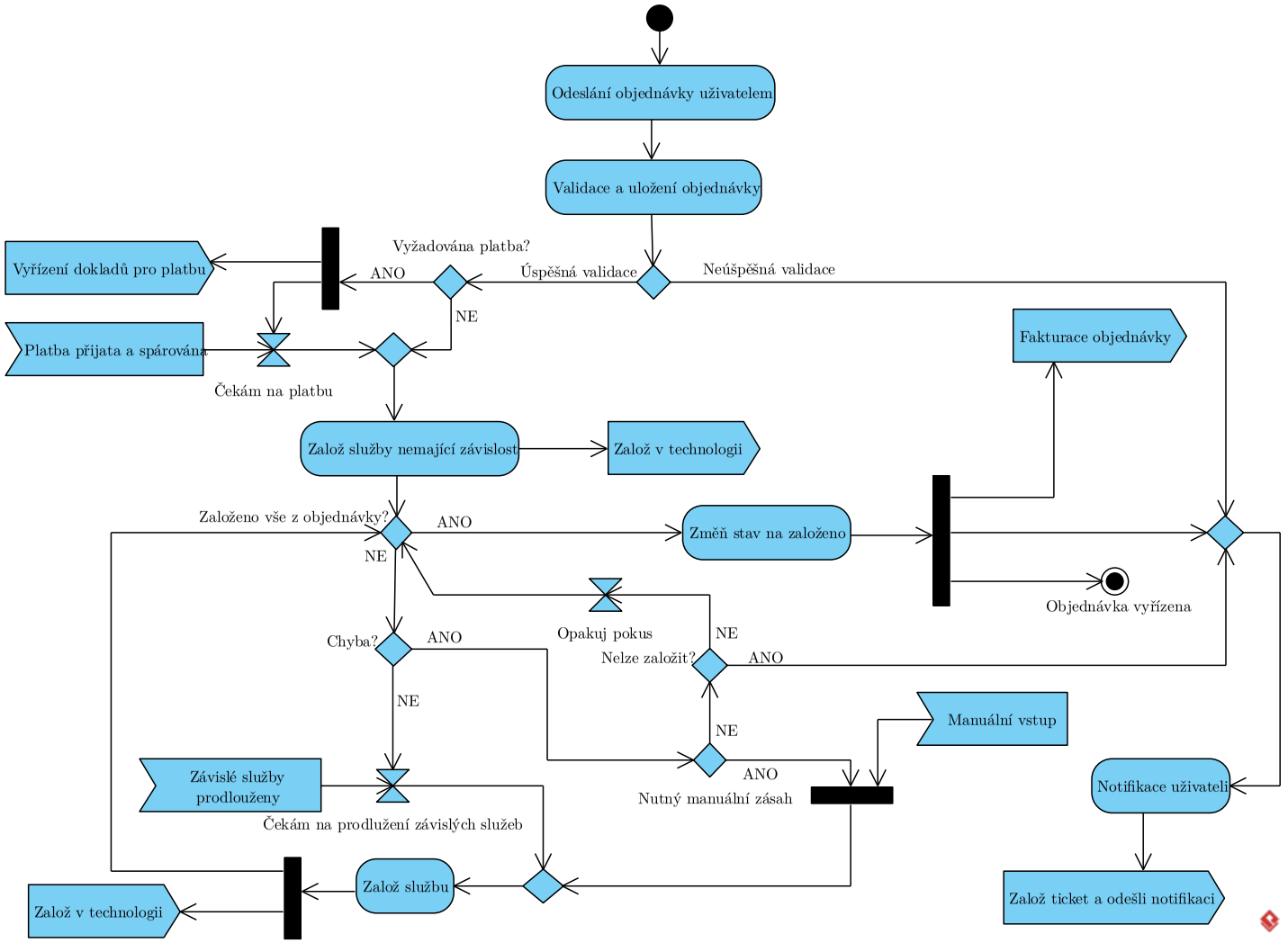
Předposledním workflow v modulu jsou procesy pro vytvoření, prodloužení a zrušení produktů obsahující konkrétní služby. Proces prodloužení a zrušení produktu obsluhuje sekvenční workflow, které pouze postupně vezme všechny služby z daného produktu a v případě prodloužení vyžádá v technologické části prodloužení. Proces prodloužení je zachycen na obrázku 4.13. V případě zrušení produktu proběhne workflow, které je mírně složitější a skládá se z těchto kroků:

1. **Ocenění zůstatkové ceny podle ceníku oprávněných nákladů** – přeplatek je vrácen zákazníkovi na uživatelský účet.

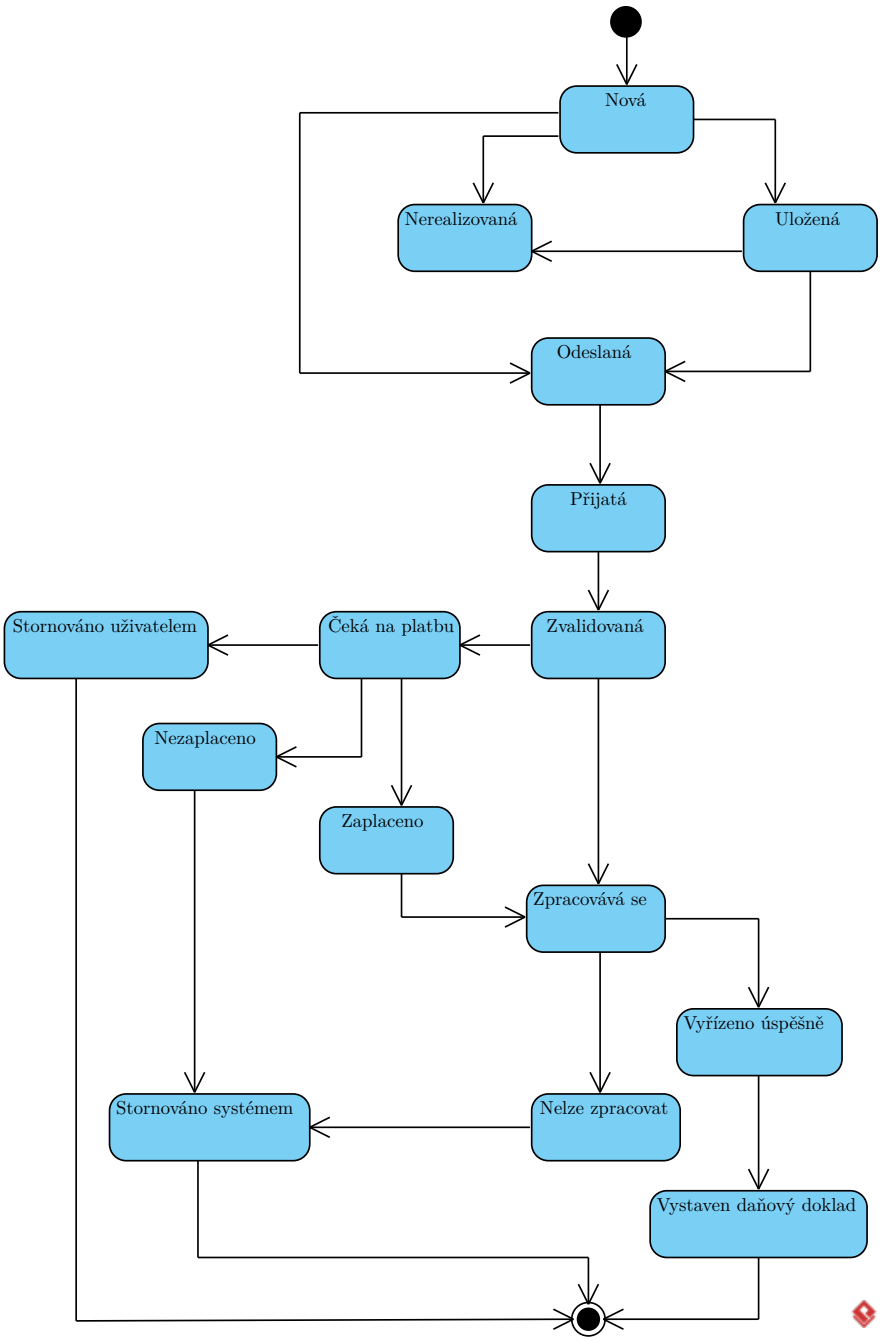
2. **Odeslání požadavku do technologické části** – ukončení všech služeb navázaných na daný rušený produkt.

Jednotlivé kroky realizované při rušení produktu jsou na obrázku 4.2.1.

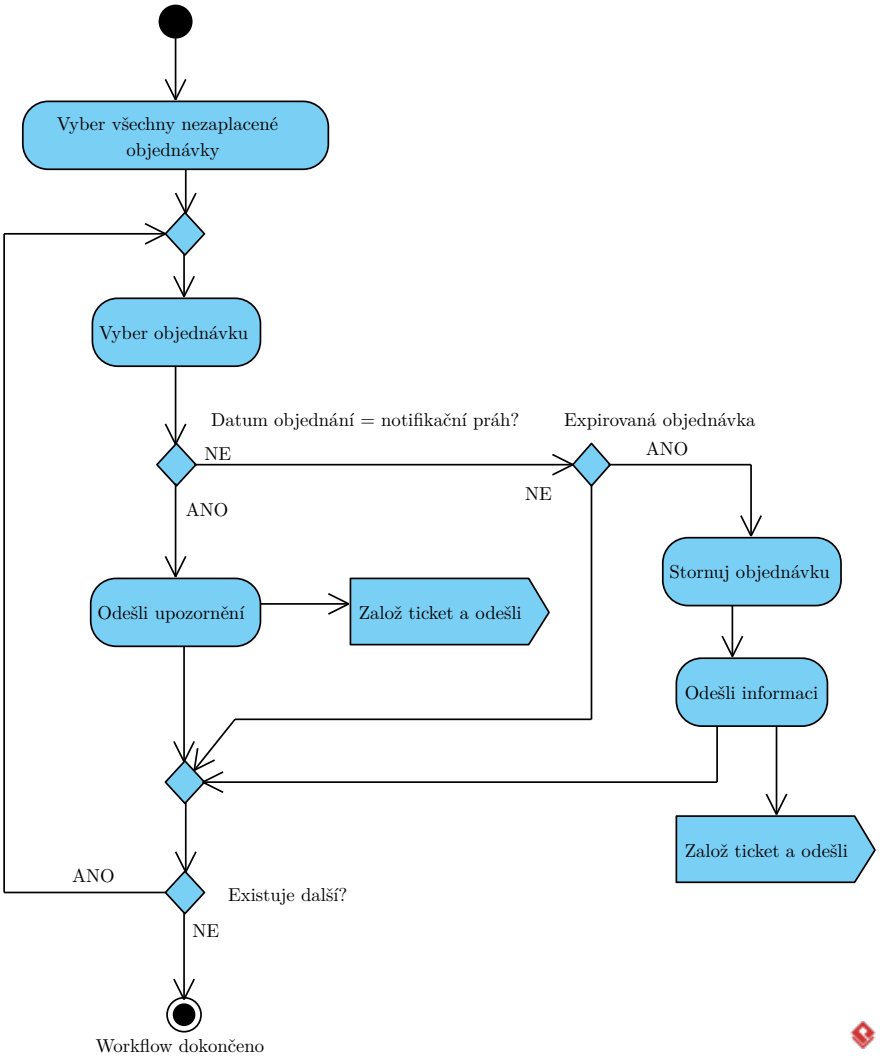
Při prodloužení a rušení produktů a jejich služeb se používá komunikační rozhraní specifikované v modulu produktů a služeb 4.1.1.



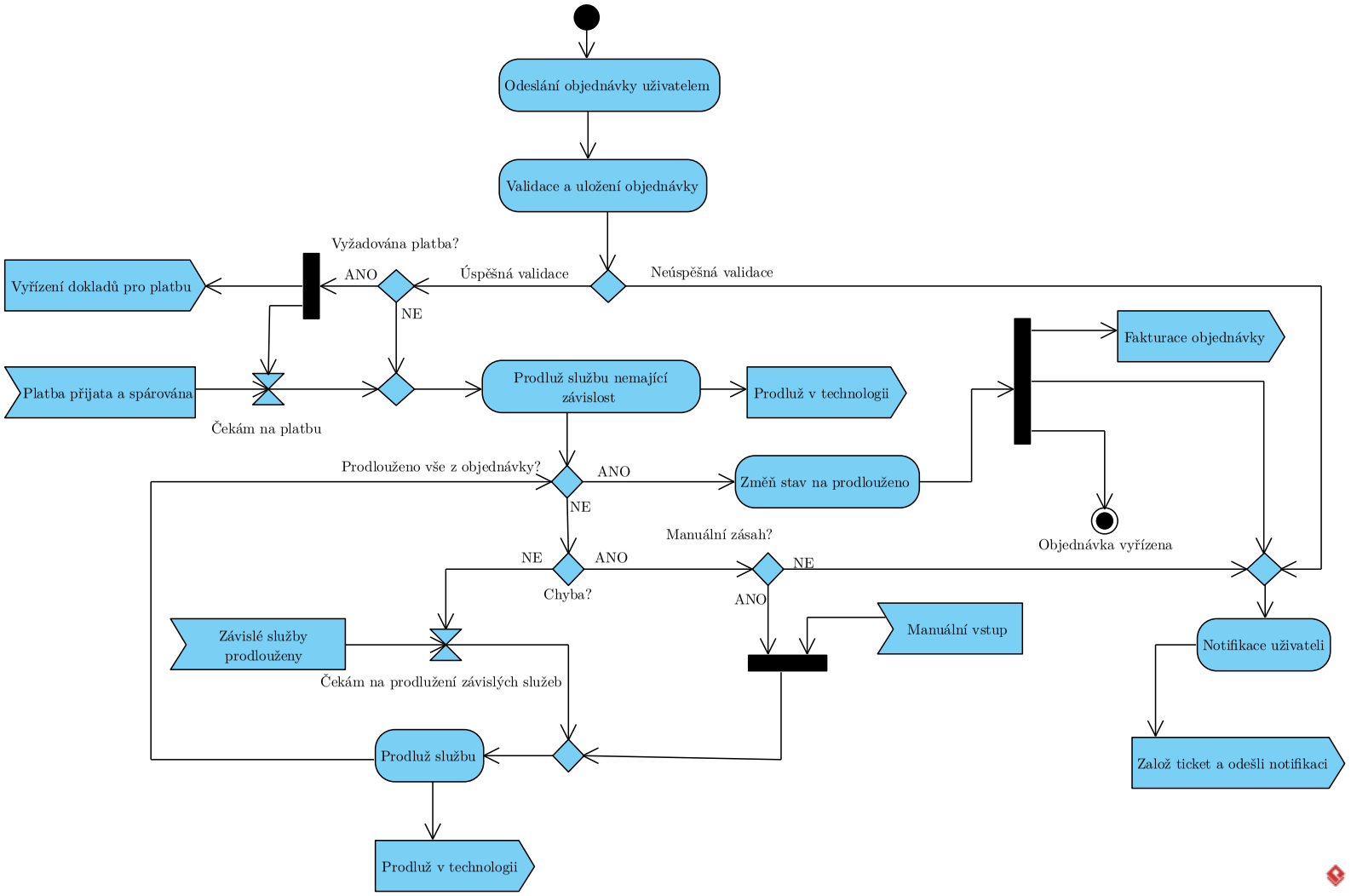

Obrázek 4.10: Proces nové objednávky



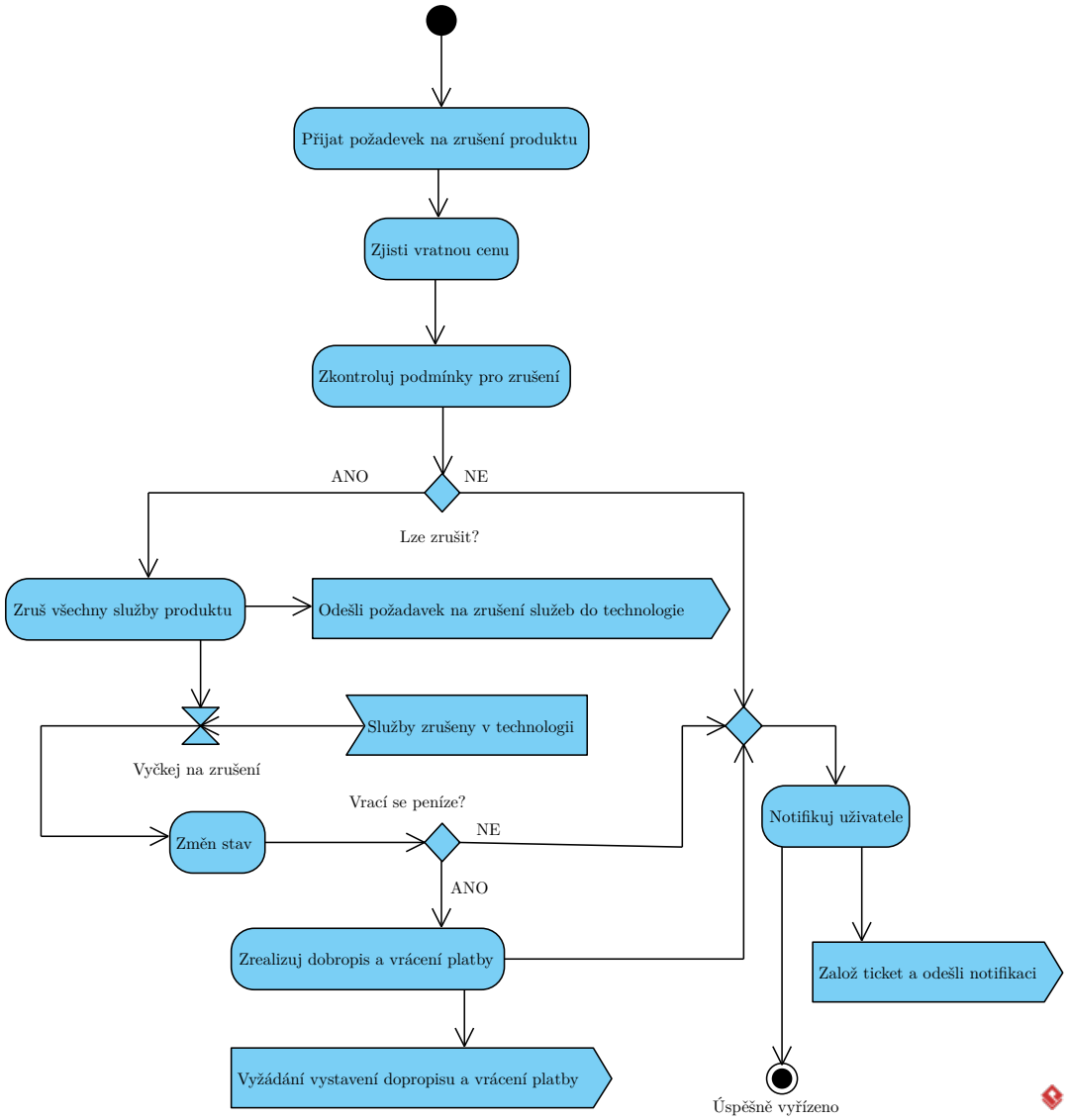

Obrázek 4.11: Stavy objednávky




Obrázek 4.12: Proces kontroly nezaplacených objednávek




Obrázek 4.13: Proces prodloužení produktu




Obrázek 4.14: Proces zrušení produktu

Dalším procesem je samotné založení služeb v technologické části po zaplacení objednávky. Toto workflow je inicializováno překlopením stavu objednávky do stavu „zaplaceno“ a probíhá následujícími kroky:

- Odeslání požadavků na založení služeb produktů, které nemají specifikovány žádnou prerekvizitu a mohou se založit přímo. Ostatní služby čekají na splnění svých prerekvizit.

- Technologická část pomocí API po úspěšném či neúspěšném provedení požadavku změní stav dané služby – mohou nastat 3 případy dalšího zpracování:

    1. **Službu nelze založit** – produkt je zrušen, peníze vráceny zákazníkovi a odeslána notifikace.
    2. **Služba je založena** – stav dané služby je změněn a tato změna spouští další zpracování. Systém projde všechny služby produktu, které ještě nebyly odeslány do technologické části. Zkontroluje jejich prerekvizity. Pokud jsou prerekvizity splněny, pošle požadavek do technologické části na zpracování dalších služeb. Tento proces se opakuje stále, dokud nejsou všechny služby založeny a objednávka produktu je úspěšně vyřízena. Samozřejmě v jakémkoliv kroku může založení služby v technologické části selhat a vrátit se k bodu 1.
    3. **Službu aktuálně nelze založit** – založení služby nelze aktuálně realizovat. Technologie zkusí založení provést později. Dále je možné opět pokračovat body 1 či 2.

- Po úspěšném založení/nezaložení služeb produktu je objednávka převedena do výsledného stavu, zákazníku odeslána notifikace o jejím vyřízení a je postoupena k fakturaci.

Poslední proces je změna nastavení parametrů produktu. Uživatel má v rámci produktů možnost provést následující změny se svými běžícími produkty:

- Změnu produktu na vyšší nebo nižší verzi dle prostupnosti daného produktu. Tato pravidla jsou nastavena v definici produktu.

- Změnu parametrů, volitelných parametrů a doplňkových služeb.

Akce popsané výše mohou mít dopad na výslednou cenu produktu. Pro další pokračování je nutné její zaplacení. Po zaplacení jsou stávající služby z produktu, které vyžadují změnu, poslány do technologické části. Služby, které aktuálně nejsou součástí, jsou zřízeny (proces je obdobný jako při zřizování nových služeb).

### 4.2.2 Servisní modul produktů (služeb)

Hlavní náplní práce modulu je podpora a automatické zpracování požadavků ohledně kontroly expirací produktů a případně jejich následné prodloužení. Tento modul je zodpovědný za další zprostředkování informací z technologické části do jiných systémů přes komunikační rozhraní. Aktuálně zadavatel nemá v tomto kontextu žádný systém, který bychom integrovali. V dalších iteracích vývoje by mohl být integrován a je nutné s ním v návrhu počítat do budoucna.

Modul je odpovědný za periodické technologické úkony předávané do technologické části. V tomto kontextu mluvíme především o:

- Zajištění běhu služeb, pokud je v rámci běhu potřeba realizovat nějakou podpůrnou akci.
- Zpracovaní procesů z fronty služeb – požadavky na zpracování přicházející z vnějšího prostředí do systému.
- Vyhodnocení stavu služby potřebné pro fakturaci. Fakturace může probíhat podle spotřebovaného výkonu či času a návazných parametrů.
- Ukončení služeb v technologické části – po nezaplacení, expiraci a dalších případech.

Servisní modul produktů poskytuje především aplikační logiku. Nemá žádné požadavky na uchovávané data v rámci systému. Všechny procesy, které zde probíhají, jsou realizovány na již uložených datech z jiných částí systému, především z modulu objednávek.

Modul realizuje následující automatické procesy v systému:

- Kontrola expirací produktů – notifikace. Proces zajišťující upozornění zákazníka na blížící se datum expirace objednaného produktu či služby.
- Kontrola expirací produktů – automatické prodloužení. Pokud má uživatel u produktu nastaveno automatické prodloužení, systém zajistí automatické prodloužení daného produktu. Prodloužení je dále navázáno na fakturaci.
- Kontrola expirací produktů – ukončení. V případě, že uživatel objednaný produkt již dále neprodlouží a není na produkt uplatněno automatické prodloužení, systém automaticky produkt po vypršení zaplaceného období zruší sám bez možnosti náhrady či vrácení zpět.
- Report dostupných produktů/balíčků – report ze systému na základě vstupního ID uživatele. Report zobrazí všechny produkty a balíčky, které má uživatel, zadaný v parametru dotazu, k dispozici. Dotaz slouží primárně pro webový front-end.

Téměř celá funkcionalita servisního modulu produktů je řešena pomocí sekvenčních workflow, které periodicky spouští interní plánovač systému. Výše zmíněné procesy z části využívají hlavní workflow pro notifikaci, prodloužení a ukončení služby definované v modulu objednávek. Schéma procesu je na obrázku 4.15. Hlavní kroky procesu jsou následující:

- Zjištění služeb v expiraci – systém se při své kontrole dotáže na všechny služby dle uživatelů a vyfiltruje je podle kritérií procesu.
- Spuštění akce – workflow iteruje po uživatelích a spouští jednotlivé akce. Při spuštění se využívá hlavní workflow, které je popsáno v modulu objednávek. Tímto workflow myslíme následující procesy:
  - Prodloužení služby
  - Ukončení služby

Navíc je modul doplněn o notifikační workflow, které informuje uživatele o různých změnách. Odeslání upozornění uživateli je realizováno pomocí komunikačního kanálu elektronické pošty a SMS. Další procesy pro zajištění běhu služeb a zpracování objednávky jsou implementovány separátně. Workflow pro zpracování procesů z fronty služeb realizuje především rozdělení požadavků podle typu a předání dalším procesům (workflow) k následnému zpracování.

Proces pro zajištění běhu služeb obsahuje také definici sekvenčních workflow podle typu realizované funkcionality. V současné iteraci zadavatel nemá k dispozici žádnou službu, která bude tímto procesem implementována. V pozdějších iteracích se s takovými službami počítá.

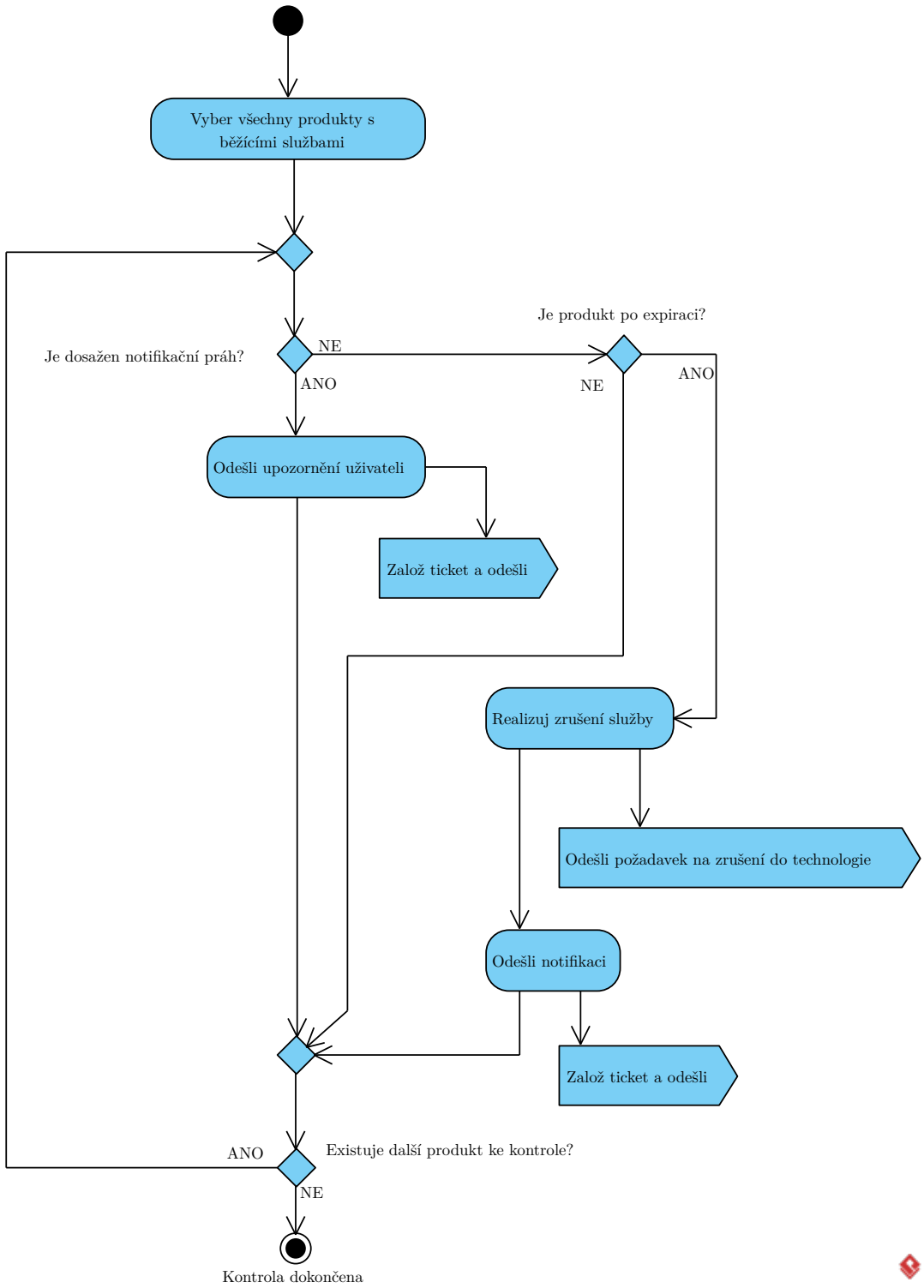
## 4.3 Blok komunikace

Hlavní náplní tohoto bloku informačního systému je komunikace se zákazníky, ať už mluvíme o komunikaci individuální (adresována přímo danému uživateli) či hromadné (především marketingová komunikace a další obchodní sdělení). Každý typ komunikace má v implementaci svá specifika a vlastní požadavky na splňovanou funkcionalitu.

Tento funkční blok poskytuje především své služby a rozhraní dalším blokům v rámci informačního systému.

### 4.3.1 Modul individuální komunikace

Tato část systému je koncepčně silně propojena s dalšími bloky systému jako je objednávkový a platební modul. Modul individuální komunikace poskytuje komunikační rozhraní po identifikaci příchozí komunikace v externím ticketovacím systému. Modul má dvě hlavní části:




Obrázek 4.15: Proces kontroly expirace produktů

1. **Komunikace** – pod touto částí rozumíme individuální komunikaci především pomocí elektronické pošty či SMS. Systém umožňuje definovat různé šablony pro individuální komunikaci pro různé účely kontaktování zákazníka. Jak bylo zmíněno výše, většina komunikace probíhá pomocí elektronické pošty. Každý emailová komunikace má založen odpovídající ticket v externím ticketovacím systému pro případ další komunikace. Systém primárně neslouží jako ticketovací systém. Pouze se v něm automaticky vytváří záznamy o provedené komunikaci, realizuje odeslání samotného emailu a dále umožňuje integraci hlavních údajů o zákazníku a jejich zobrazení přímo v externí aplikaci

2. **Rozhraní pro identifikaci klíčových údajů** – druhá část modulu obsahuje především sadu komunikačních rozhraní integrovaných do nového externího ticketovacího systému. Rozhraní poskytuje na základě hlavních identifikátorů informace o zákazníkovi a související data, ke kterým se ticket váže.

Modul je využit pro zasílání následujících informací:

- Výzvy k platbě, zálohového či daňového dokladu.
- Oznámení o zřízení, změnách či zrušení služeb.
- Informace o objednávce a jejím stavu.
- Individuální nabídky – nabídka pro zákazníka spíše technického rázu, nejedná se marketingovou komunikaci.
- Informace o existenci nerealizované objednávky.
- Požadavek na autorizaci služby.
- Realizace požadavku na změnu hesla.

Modul při realizaci každé komunikace automaticky zakládá danou komunikaci v ticketovacím systému přes jeho API. Jedná se o založení odesílané komunikace a její automatické odeslání koncovému příjemci. Není vyžadována žádná další služba či proces provádějící další operaci. Založení ticketu v externím systému je vyžadováno z důvodu další následné komunikace se zákazníkem. To umožní pracovníkovi technické podpory zobrazení celého komunikačního vlákna, včetně úvodní zprávy, která obsahuje hlavní a stěžejní informace celé komunikace.

Komunikační API, které používá externí ticketovací systém, poskytuje na základě vstupních informací následující data o zákazníku, jeho objednávce a službě:

- Údaje o zákazníkovi (clients/name/GUID).

- Údaje o fakturačním kontaktu (clients/invoicecontact/GUID).
- Informace o objednávce (clients/order/GUID).
- Informace o službě (clients/service/GUID).

Další možnosti API budou rozšířeny dle požadavků zadavatele v dalších iteracích vývoje systému.

### 4.3.2 Modul hromadné komunikace

Tento modul oproti předchozímu zajišťuje marketingovou komunikaci. Funkcionality modulu dále nerozšiřují již existující funkcionalitu navržených částí. Pouze komunikují s modulem zákazníků, objednávek a fakturace z důvodu sestavení listu příjemců. Modul je primárně určen pro marketing a hromadné oslovení zákazníků.

Jeho hlavní požadované vlastnosti jsou následující:

- Správa šablon – v tomto kontextu myslíme šablonu zprávy určenou k hromadnému oslovení zákazníků. Primární komunikační kanál se předpokládá pomocí elektronické pošty. Se šablonami musí být umožněna základní práce jako je vytvoření, změna či smazání.
- Vytvoření seznamu zákazníků – vytvoření seznamu kontaktů k hromadnému oslovení podle předem předvolených filtrů a nastavení.
- Hromadné oslovení zákazníků – hromadné oslovení zákazníků spojením vybrané šablony a seznamu kontaktů. V textu může být dále popisována také jako „Vytvoření kampaně“.
- Odhlášení klienta ze zasílacího seznamu – nutný požadavek ke splnění české legislativy.
- Výběr komunikačních kanálů – nastavení a povolení vybraných komunikačních kanálů u jednotlivých kontaktních osob. Nastavení bude provádět ve svém profilu přímo koncový uživatel (zákazník).

U tohoto modulu se přímo počítá, že marketingová komunikace bude probíhat přes externí systém, který je na tuto problematiku úzce orientován. V navrhované části informačního systému se zabývám pouze vytvořením a editací marketingových šablon a dále také seznamu zákazníků. Zadavatel aktuálně počítá s využitím marketingového nástroje „MailChimp“, který se přímo orientuje na takovouto marketingovou komunikaci. V informačním systému je evidována pouze informace o kampaních kvůli jednoduššímu přehledu. Samotné zpracování a odeslání kampaně probíhá pomocí systému Mailchimp. Komunikace a synchronizace dat probíhá pomocí REST API systému Mailchimp přímo z informačního systému.

V modulu jsou evidovány pouze následující informace:

- Marketingový seznam zákazníků – pro evidenci seznamu je využita již existující funkcionalita v rámci použitého frameworku, která s sebou nese specifikaci seznamu a podle něj zobrazí (vybere) odpovídající kontakty. Nastavení specifikace seznamu je přímo v kompetencích obsluhy systému.

- Šablony marketingové komunikace – pro uložení těchto dat je použita stávající entita rozšířená o položky specifické pro systém Mailchimp.

- Historie kampaní – opět je využita funkcionalita dodávaná v rámci frameworku, která zajišťuje při vytvoření nové kampaně její evidenci. V historii jsou uchovávány základní informace o kampani a seznam klientů, kterým kampaň byla zaslána.

- Rozšíření uživatelského nastavení – pro uchování informací o nastavení komunikačních kanálů v rámci jednotlivých uživatelských účtů byla ke každému účtu přidána další entita se seznamem dostupných komunikačních kanálů a jejich povolení či zakázání pro komunikaci.

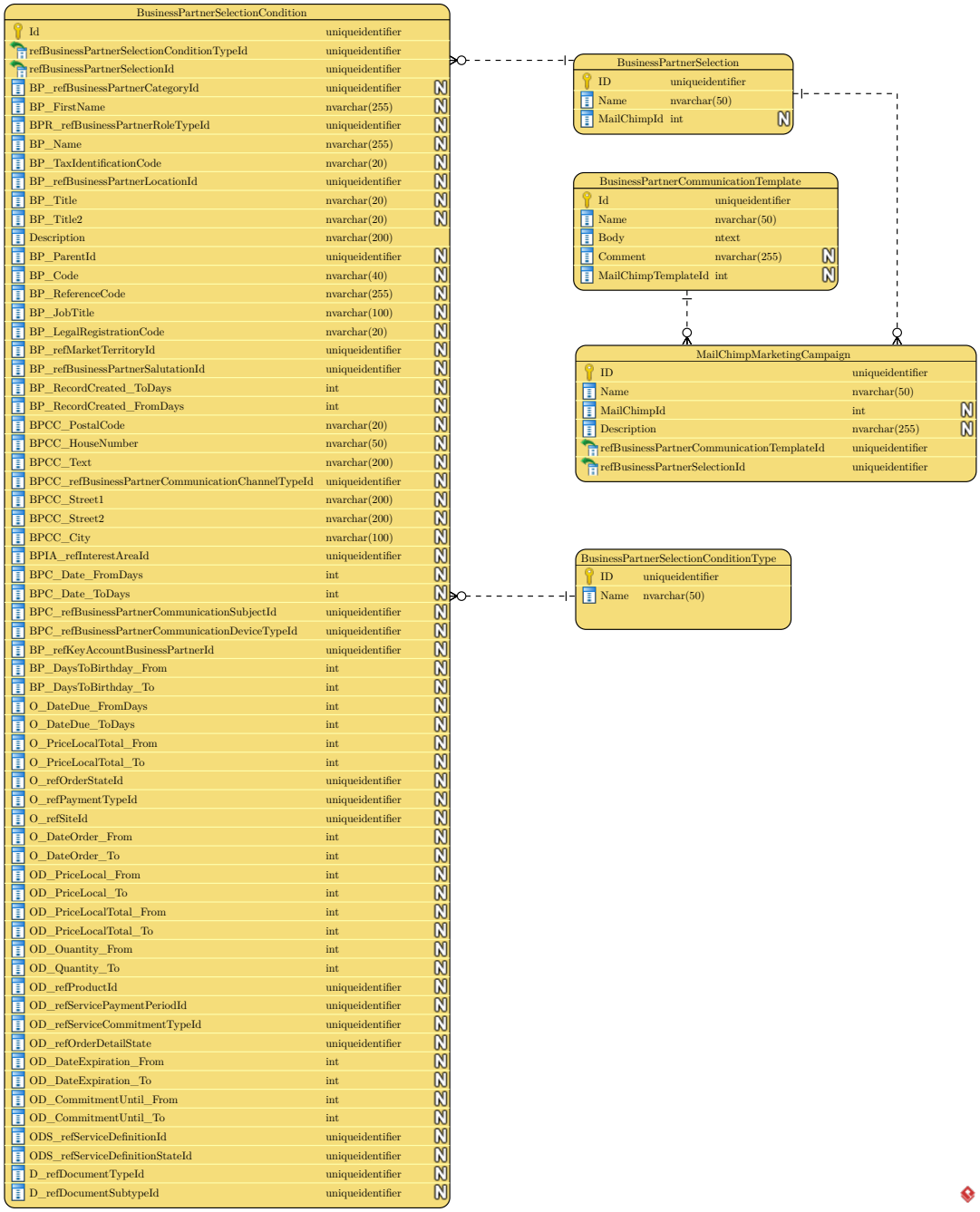
Navržený datový model modulu je zobrazen na obrázku 4.16.

Jak již bylo výše v textu zmíněno, modul bude velmi úzce provázán s aplikací Mailchimp pomocí jejího REST API. V rámci propojení se jedná především o práci se zasílacími seznamy, šablonami komunikací a samotnými kampaněmi, které budou dále přes aplikaci distribuovány koncovým uživatelům ve formě emailů. Integrace informačního systému s aplikací Mailchimp je provedena v následujících úrovních:

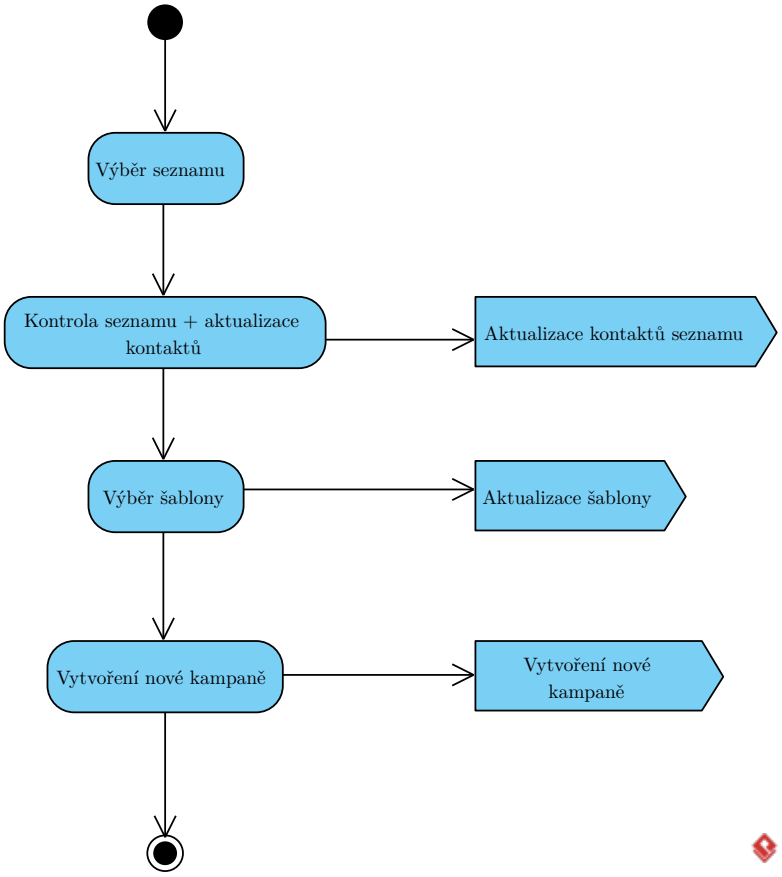
- Vytvoření nového zasílacího seznamu – seznam po vytvoření v informačním systému je také automaticky vytvořen v Mailchimpu. V informačním systému je u seznamu uložen jeho externí identifikátor, kterým ho lze jednoznačně identifikovat v Mailchimpu. Zároveň jsou zde promítány veškeré změny v nastavení zasílacího seznamu – změna názvu a další základní vlastnosti.

- Přiřazení kontaktů pod zasílací seznam – po vytvoření zasílacího seznamu jsou pod seznam v Mailchimpu přiřazeny kontakty odpovídající kritériím z daného seznamu. Pokud je zasílací seznam aktualizován, jsou kontakty upraveny a provádí se změna základních evidovaných údajů v Mailchimpu. Pokud kontakt už kritériím zasílacího seznamu nevyhovuje, je smazán.

- Marketingová šablona – při vytvoření je zároveň šablona vytvořena v aplikaci Mailchimp. Její identifikátor z aplikace Mailchimp se pro jednoznačnou identifikaci uloží také do informačního systému. Samozřejmostí je automatická aktualizace šablony v aplikaci při úpravě v systému.




Obrázek 4.16: Datový model hromadné komunikace

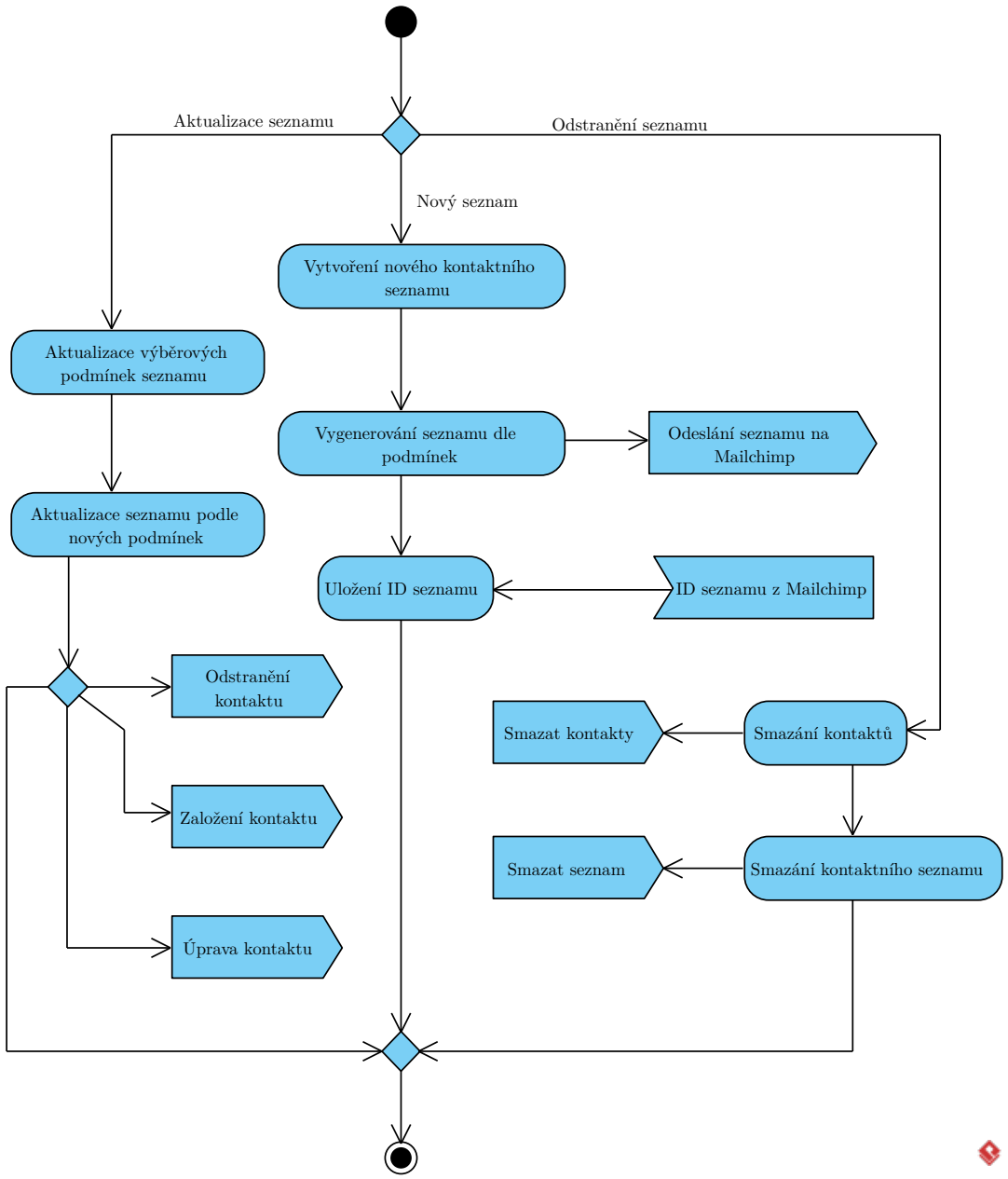



Obrázek 4.17: Vytvoření nové kampaně

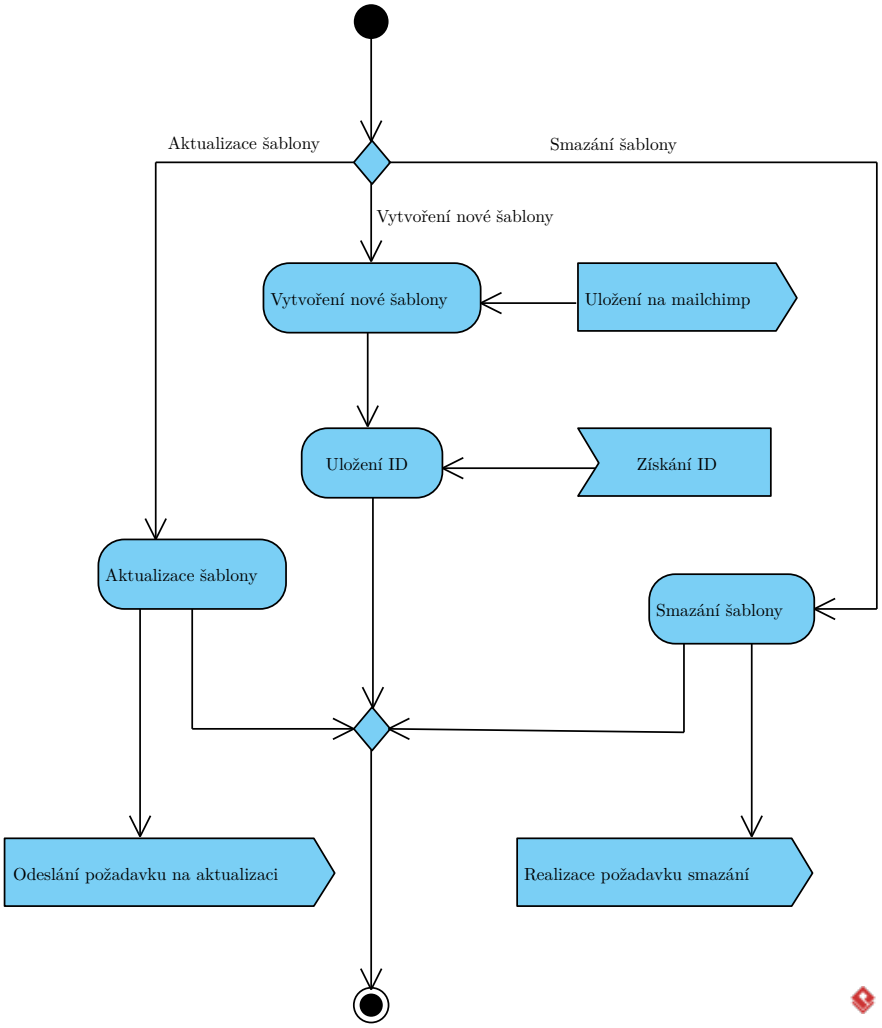
- Marketingová kampaň – v obecném kontextu se jedná o spojení šablony se zasílacím seznamem. Při vytvoření kampaně v informačním systému je kampaň automaticky vytvořena v aplikaci Mailchimp a následně rozeslána zákazníkům. V rámci vytvoření kampaně je před jejím odesláním ještě zvalidován seznam příjemců, zda se nezměnil, a případně proběhne aktualizace. Tím je zajištěno odeslání vždy všem zákazníkům, kteří splňují konkrétní kritéria zasílacího seznamu. Proces vytvoření marketingové kampaně je na obrázku 4.17.

Proces založení nového seznamu uživatelů je zobrazen na obrázku 4.18 a postup při založení nové šablony na obrázku 4.19.

Implementovaný informační systém neřeší část „Odhlášení klienta ze zasílacího seznamu“. Tato funkcionalita je realizována přímo v rámci aplikace Mailchimp. Při synchronizaci jednotlivých kontaktů v rámci zasílacích seznamů se tato položka nemění a nepracuje se s ní. Přihlášení klienta k zasílacímu seznamu je zajištěno pomocí nastavení u jeho uživatelského účtu –




Obrázek 4.18: Vytvoření a úprava nového zasílacího seznamu




Obrázek 4.19: Vytvoření a úprava šablony komunikace

povolí pro sebe vybrané komunikační kanály, ze kterých chce marketingové informace přijímat. Toto nastavení se také promítá při vytváření zasílacího seznamu.

## 4.4 Blok správy zákazníků

Tato část navrhovaného systému poskytuje funkcionality pro evidenci zákazníků, jejich kontaktních údajů, kontaktní osoby, přiřazení do skupin, fakturační kontakty a další známé položky ze světa CRM systémů. Dále obsahuje informace o uživatelských účtech, protože zde máme rovnost uživatelský účet = kontaktní osoba (fakturační kontakt).

Blok poskytuje pouze uchování a ověření přihlašovacích údajů k jednotlivým účtům v systému. Nestará se o vystavování autorizační tokenů a funkcionality spojení se Single Sing-on (SSO). Funkcionalita SSO bude realizována externím modulem aplikace, na který se tento návrh nevztahuje.

### 4.4.1 Modul správy zákazníků

Tento důležitý modul informačního systému plní především funkci adresáře firem a kontaktů. Nejsou zde implementovány žádné speciální a rozšiřující funkcionality. Modul má primárně k evidenční funkci. Očekáváme od něj následující funkcionalitu:

- Správa zákazníků – možnost vytvoření nového fakturačního kontaktu a správa kontaktu. Evidují se základní známé položky jako v CRM systémech. Pod jedním obchodním partnerem může být více fakturačních adres (firma provádí outsourcing služeb pro své klienty)
- Kategorie zákazníků – možnost přiřazení různých kategorií danému obchodnímu partneru.
- Nastavení práv kontaktních osob – vzhledem k faktu, že kontaktní osoba, může být uživatel, je vyžadováno mít možnost nastavení práv pro jednotlivé kontaktní osoby.
- SuperAdmin – možnost zvolit administrátora pro daného obchodního partnera.

Pokud se blíž podíváme na datový model, modul uchovává následující informace:

- Obchodní partner – základní údaje o obchodním partnerovi, IČO, DIČ a další
- Kontaktní osoby – kontaktní osoby a základní informace o nich. Vždy se vážou ke konkrétnímu obchodnímu partnerovi.
- Kontakty – kontakty se rozlišují podle svého druhu (adresa, telefon, email a další). Je možné je definovat zvlášť pro samotného obchodního partnera a pro kontaktní osoby obchodního partnera. V rámci toho je možné i definovat nastavení pro příjem technických a marketingových sdělení pro každý kontakt. U kontaktů je možnost nastavení položky superadmin (samozřejmě pouze pokud má uživatel odpovídající roli).
- Kategorie – přiřazené kategorie k danému obchodnímu partnerovi
- Ceníky – přiřazené ceníky vázané k danému obchodnímu partnerovi. Definováno v modulu ceníku a slev 4.1.3.

- Role – opět možné definovat pro obchodního partner či kontaktní osobu. U kontaktní osoby se tímto omezují práva pro přístup.

- Poznámky – poznámky technického či obchodního rázu pro obsluhu systému či marketing. Neveřejná položka.

Při zakládaní obchodního partnera je možné využít načtení ze systému ARES podle IČO. Systém vrátí vyplněné základní informace o obchodním partnerovi jako název a kontaktní adresu. Uživatel je má samozřejmě před uložením možnost editovat. K jednotlivým položkám adresáře se velmi úzce váže historie proběhlé komunikace, kterou je možné zobrazit a procházet.

Detailní datový model tohoto modulu je na obrázku 4.20.

Pro implementaci adresáře je využit již existující adresář kontaktů v rámci frameworku, který je rozšířený o specifické položky definované zadavatelem. Především se jedná rozšíření o ceníky a role uživatelů.

### 4.4.2 Modul správy uživatelských účtů

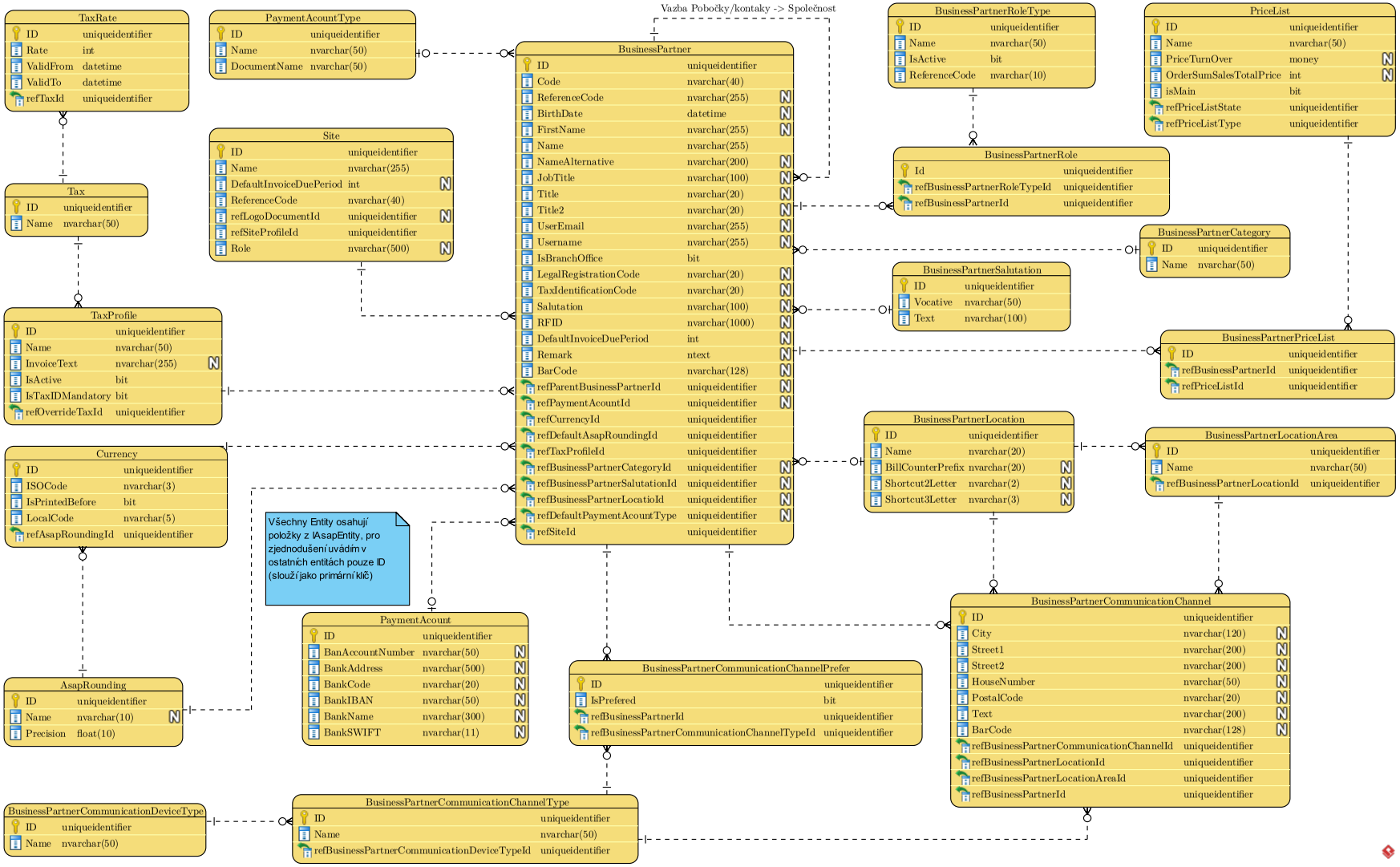
Modul správy uživatelských účtů se stará o uchování přihlašovacích údajů jednotlivých uživatelů nutných pro přístup do systému. Mezi uživatele řadíme jak interní zaměstnance zadavatele jako operátory systému, tak externí uživatele – zákazníky zadavatele. Z pohledu těchto dvou skupin musí být uživatelé nějakým způsobem odděleni, nehledě na to, že každá skupina bude využívat jiný způsob přihlašování. Pro interní uživatele se předpokládá vyžití přihlašování pomocí uživatelského jména a hesla ze služby Active Directory. U externích uživatelů je využita interní datová databáze uživatelů s jejich uživatelskými jmény a bezpečně uloženými hesly k účtům.

Postup pro založení nového externího uživatele s navázáním na fakturační kontakt je následující:

1. Založení externího uživatele (buď přímo daným uživatelem přes webové rozhraní či technickou podporou) – pod založením uživatele rozumíme vyplnění základních vyžadovaných informací o uživateli jako je jméno, email a další standardní reálie.

2. Po vytvoření účtu má uživatel možnost založit nového obchodního partnera (popsáno v kapitole 4.4.1). Poté se automaticky stává super administrátorem tohoto obchodního partnera a je přidán do jeho kontaktních osob. Může dále vytvářet a přiřazovat další uživatele.

Uživatelské účty mohou zakládat buď přímo externí uživatelé přes web, či mohou být založeny technickou podporou. Uživatelské účty interních zaměstnanců zakládá určený správce informačního systému. Jak je patrné z výše popsaného postupu založení účtu, informace o daném uživateli, jako je email a přihlašovací jméno, jsou uloženy v databázi přímo u kontaktní osoby. Pro




Obrázek 4.20: Datový model modulu zákazníků

externí uživatele ještě existuje další entita, která obsahuje jejich přihlašovací údaje, ověření a heslo.

Datový model modulu je na obrázku 4.21.

Při implementaci modulu je využito částí frameworku, které pokrývají správu uživatelských účtů. Modul uživatelských účtů funkčně pokrývá tyto vyžadované činnosti:

- Založení účtu
- Deaktivace účtu
- Zablokování účtu (buď technickou podporou či po několikátém neúspěšném pokusu o přihlášení)
- Ověření emailové adresy
- Odblokování uživatelského účtu

Dále je modul ještě doplněn o ověření telefonního čísla pomocí autorizační SMS zprávy.

Modul je velmi úzce propojen s modulem pro správu zákazníků v kapitole 4.4.1. Z částí využívá stejné datové struktury pro uložení informací o založených uživatelích. Ve své podstatně každý založený uživatel je zároveň při vytvoření obchodní partner. Při vytvoření vlastního obchodního partnera se stává jeho kontaktní osobou. Nedá se přímo říci, že uživatel = obchodní partner (kontaktní osoba), ale spíše uživatel je podmnožinou obchodních partnerů (kontaktů). U obchodních partnerů mohou samozřejmě existovat kontaktní osoby, které nemají v systému založený uživatelský účet.

## 4.5 Blok finančních operací

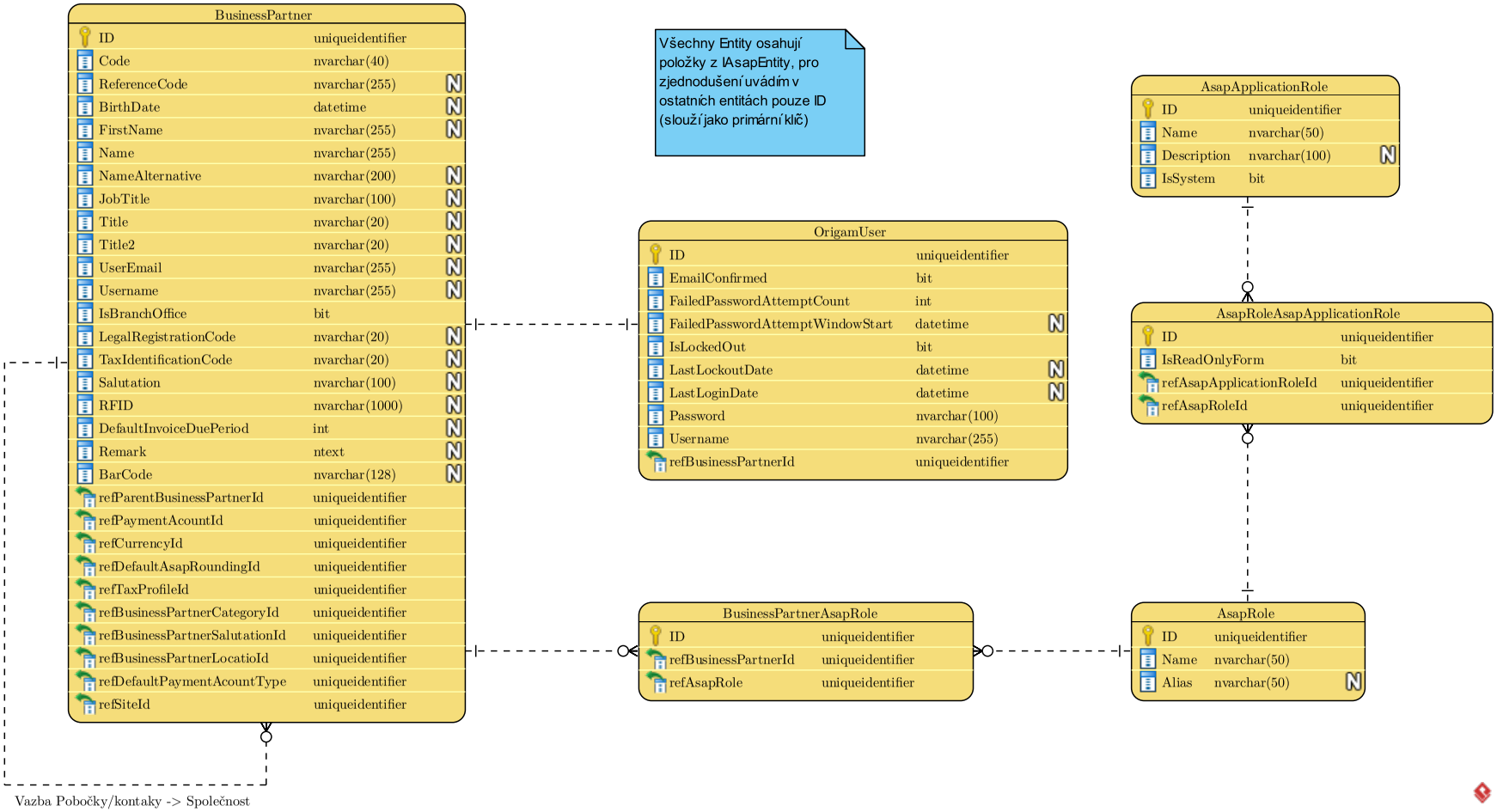
Hlavní funkční náplní posledního bloku je zpracování a vystavování faktur, výzev k platbě, párování plateb a další operace spojené s agendou faktur, výzev a plateb za objednané služby.

### 4.5.1 Fakturační modul

Jak z názvu modulu vyplývá, primárně se stará o vystavování faktur, výzev k platbě a párování plateb. Ze své povahy nebude realizovat žádné účetní operace. Všechny doklady vzniklé v tomto modulu budou exportovány do externího účetního programu, ve kterém bude vše zpracováváno.

Nyní se podívejme blíže na funkcionality, které tento modul musí splňovat:

- Vystavení daňového dokladu či vystavení zálohového dokladu
- Vystavení výzvy k platbě za objednané služby




Obrázek 4.21: Datový model uživatelských účtů

- Upomínání platby (možnost sjednocení upomínání na jednotné datum)
- Možnost zaplacení dokladu pomocí platebního modulu vyžádáním inkasa z karty
- Párování plateb (zde je uvažováno primárně podle variabilního symbolu, případně podle dalších atributů) - párování plateb poběží standardně automaticky, modul ovšem musí umožňovat i manuální spárování platby pokud dojde k problému či to bude aktuální případ vyžadovat.
- Operace s volnými zálohami – speciální funkce, která umožňuje mít v rámci účtu zákazníka k dispozici volné finanční prostředky pro možné platby. Jedná se o obdobu zálohového účtu, který nabíjíme pomocí finančních prostředků. V rámci těchto volných záloh musí systém umožňovat:
  - Párování volné zálohy – stejná analogie jako u párování plateb. Provázanost se předpokládá pomocí variabilního symbolu, v dalších iteracích může proběhnout implementace dalšího identifikátoru pro párování platby
  - Uvolnění volné zálohy
  - Přijetí neadresované platby a její identifikace – tím myslíme přijetí platby a uložení jako volné zálohy v rámci zákaznického účtu. Identifikace probíhá znovu na základě variabilního symbolu či jiného identifikátoru.
- Vystavení dobropisu
- Vratka peněz – zde rozumíme realizaci zpětné platební operace směrem ke klientovi
- Informování klienta o proběhlých platbách a jejich stavech
- Přenos dokladů do účetního systému

Hlavním výstupem fakturačního modulu jsou generované doklady ve formátu PDF. Jakmile dojde k vystavení dokladu, již na něm nelze provést žádnou změnu – doklad je neměnný, resp. data využitá pro vygenerování dokladu se ihned zamykají, aby jej nebylo možné změnit. Abychom zajistili věrohodnost vystavovaných dokumentů, každý dokument je ihned po vygenerování odesílán do externího systému, kde je opatřen zaručeným elektronickým podpisem. Po navrácení elektronicky podepsaného dokumentu, je dokument uložen v systému k odpovídajícímu záznamu (objednávce), ke které se doklad váže. Při jakémkoliv dalším požadavku na zobrazení dokumentu k danému záznamu je využit uložený dokument, neprobíhá nové generování (a podepisování). Tyto elektronicky podepsané dokumenty – jsou dále odesílány koncovým

příjemcům, tedy objednavatelům služeb (nebo jejich nastaveným fakturačním kontaktům).

Celý modul je velmi úzce propojen s dalšími komponentami systému, které dohromady realizují celý objednávkový proces – od objednání, zřízení služby, fakturace, vyřízení platby až po komunikaci se zákazníkem. Modul čerpá data z těchto komponent:

- Objednávkový modul – informace o objednávce a objednaných službách
- Správa zákazníků – kontaktní informace, fakturační údaje, vazba na objednávkový modul

Modul využívá služby těchto komponent systému:

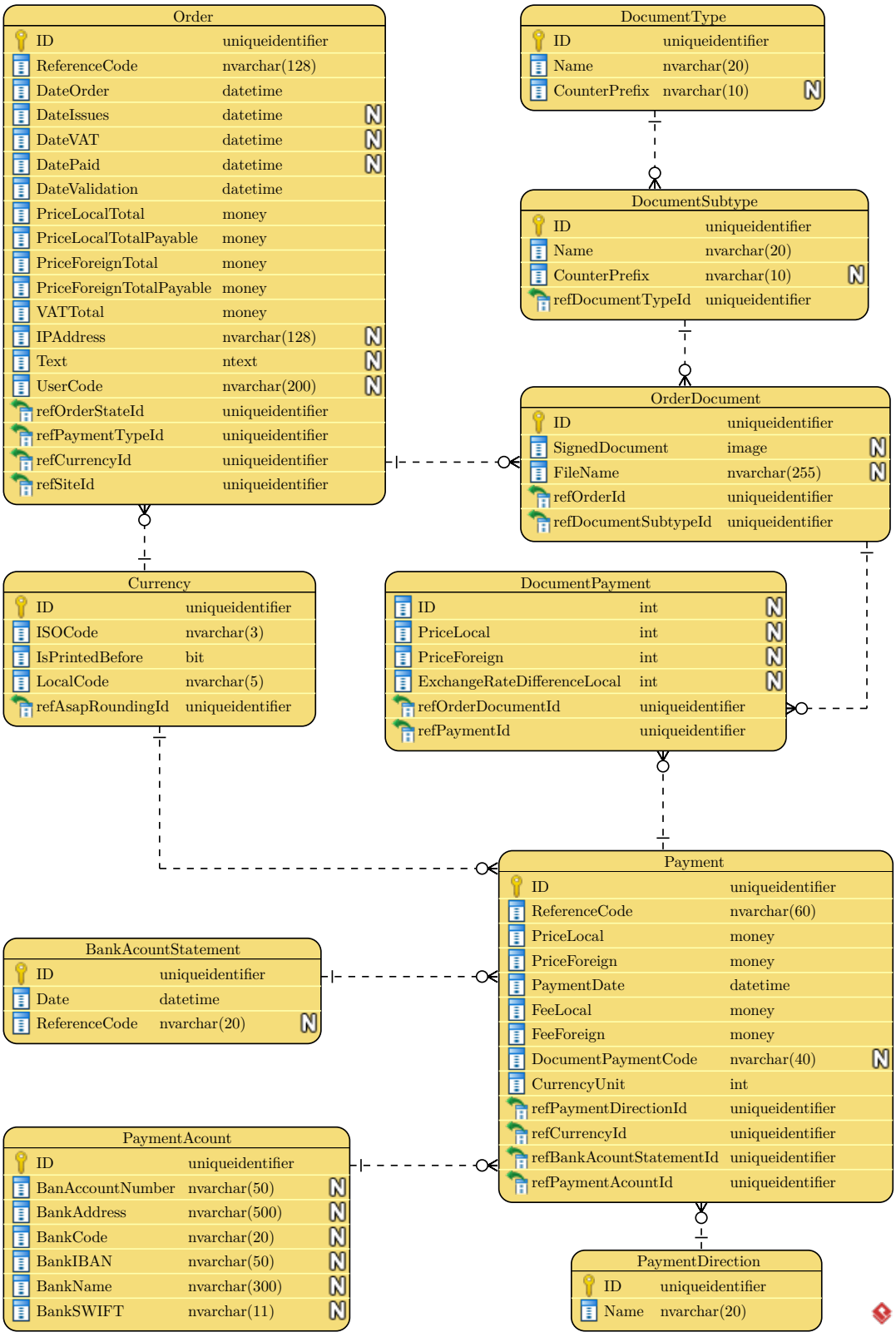
- Modul individuální komunikace – přes něj odcházejí veškeré notifikace, informační emaily a další zprávy nutné pro úspěšné vyřízení objednávky. Možných komunikačních kanálu může být více. Primárně je aktuálně plánováno využití komunikace pomocí elektronické pošty.
- Platební modul – požadavek na vyřízení platby za služby pomocí platební karty

Protože můžeme říci, že fakturační modul ve své podstatě tvoří nadstavbu a rozšiřuje objednávkový modul, mají oba moduly velmi úzce také propojené datové modely. Pokud půjdeme do detailu, tak fakturační modul pouze rozšiřuje datový model modulu objednávek.

Celé propojení vychází ze základu, že hlavní datový zdroj pro fakturaci jsou objednávky. Modul je rozšířen o úložiště vygenerovaných podepsaných dokladů. Dokumenty jsou ukládány v jedné entitě a ke každému dokladu je připojen záznam, k jaké objednávce se váže, a jeho typ. Rozšířený datový model je na zachycen a popsán na obrázku 4.22.

V rámci modulu je realizováno automatické zpracování některých procesů. Mezi tyto procesy patří:

- Automatické vygenerování dokladů, jejich odeslání k elektronickému podepsání a následné uložení do systému – všechny tyto procesy jsou prováděny většinou při změně stavu objednávky.
- Komunikace s klientem - automatické odeslání dokladů či případně upozornění na změny stavu objednávky.
- Automatické stahování plateb z banky a jejich identifikace a párování.
- Vyžádání platby u platebního modulu, pokud je povoleno automatické vyřízení či prodloužení.




Obrázek 4.22: Datový model fakturačního modulu

### 4.5.2 Platební modul

Hlavní náplní práce modulu je realizace plateb pomocí platební brány. Ve své podstatě slouží k těmto dvěma funkcím:

- Vyžádání platby na externím platebním modulu (zde uvažujeme především automatické inkaso z platební karty)
- Nadstavba pro fakturační modul – propojovací rozhraní mezi externím platebním a fakturačním modulem a předání informací o provedené platbě fakturačnímu modulu

Modul nerozšiřuje dále datový model celého informační systému, pouze přidává komunikační rozhraní do externího systému.

Kapitola 5

# Prototypování

V předchozí kapitole byl proveden návrh všech modulů informačního systému. Návrh se týkal především datové vrstvy, kritických procesů a zamýšlených komunikačních rozhraní pro integraci s dalšími systémy. Ověření realizovaného návrhu lze provést pomocí formální verifikace, simulací či můžeme přímo realizovat prototyp informačního systému.

Protože je pro realizaci využit framework Origam, který je zástupce MDA, už z podstaty této věci se jako nejvhodnější varianta pro ověření jeví realizace prototypu. V rámci prototypu jsem ověřoval hlavně správnost navržené datové vrstvy pro uložení dat a dále především komunikační rozhraní systému, která byla dostupná během realizace prototypu u klienta. Podrobnějším detailům vývoje prototypu je věnována další sekce této kapitoly v části 5.2. Nejprve rozeberu samotné prostředí, ve kterém probíhala implementace informačního systému.

## 5.1 Vývojové prostředí

Framework Origam běží jako služba v rámci webového serveru, je potřeba jednotlivé vývojářské instance od sebe vhodným způsobem oddělit. Pro naše potřeby existují tyto 3 vývojové instance aplikačního serveru:

- **Vývoj** – primárně vývojová verze, probíhá nad ní celý vývoj a obsahuje vždy nejnovější verzi systému. Nachází se zde vždy verze N+1.
- **Test** – testovací verze, přechází zde verze N z vývoje, jejíž iterace byla již ukončena. Slouží k testování implementovaných funkcionalit a celého systému. Prostředí je postaveno tak, že slouží pro vývojáře (testery) i pro testy ze strany zákazníka.
- **Produkce** – poslední prostředí, kde je nasazena již verze N-1, která sem přechází z testu. Tato verze je považována za stabilní a otestovanou. Lze ji použít přímo pro produkční provoz, demo či pilotní provoz.

Abychom dostatečně eliminovali možné dopady a výkonové problémy ve vývojovém a testovacím prostředí, jsou obě prostředí oddělena od produkčního prostředí. Oddělení je provedeno na úrovni virtuálních strojů, na kterých jednotlivé prostředí běží. Pro provoz nestačí oddělení na úrovni aplikačního serveru – zde by právě mohlo docházet k problémům, například při spuštění nekonečné smyčky ke zpracování dat. Rozdělení aplikačních serverů na virtuálních strojích je následující:

- Vývoj a test
- Produkce

V našem vývojovém prostředí je použit aplikační webový server IIS od společnosti Microsoft.

Celé vývojové prostředí ještě doplňuje další klíčový prvek infrastruktury – virtuální stroj s databázovým serverem. Na tomto stroji jsou spuštěny 2 instance SQL serveru od společnosti Microsoft. Dvě instance jsou využity opět kvůli oddělení jednotlivých databází sloužících pro jednotlivé prostředí. Databáze pro aplikační server jsou do instancí rozděleny následovně:

- Vývoj a test
- Produkce

Rozdělení na instance v tomto případě stačí, protože jednotlivým instancím jsme velmi dobře schopni definovat maximální využití systémových prostředků. Eliminujeme tak možné výkonové problémy a jejich možnost přenosu z instance pro vývoj a test.

Pro každou vývojovou instanci existují dvě samostatné databáze:

1. **Model** – model aplikace, který je pak interpretován Origam enginem ve spustitelnou webovou aplikaci
2. **Data** – persistentní úložiště dat pro aplikaci. Jedná se o databázi vytvořenou při vývoji, které obsahuje veškerá provozní data a konfigurace.

### 5.1.1 Verzování

Pro snadnou udržitelnost vývoje informačního systému jsou všechny zdrojové kódy verzovány. Jedná se o verzování na dvou úrovních:

1. **Verzování balíčků** – specifická část vycházející z frameworku. Každý balíček má vždy seznam realizovaných verzí a jednu aktuální verzi. Postupnou aktualizací od nejnižší verze balíčku po aktuální dosáhneme aktuální verze modelu aplikace. Poslední aktuální verze konkrétního balíčku je uložena v datové databázi.

2. **Verzování celého modelu prototypu** – jedná se o verzování celého modelu aplikace. Vývojové prostředí zapisuje jednotlivé změny v modelu do souborů odpovídajícího balíčku. Jedná se o export konkrétní části modelu do formátu XML, který lze velmi dobře verzovat. Pro naše potřeby bylo využito verzovacího systému GIT se vzdáleným repozitářem na serveru bitbucket.org.

Druhý typ verzování je důležitý především při přenosu aplikace z jednoho produkčního prostředí do dalšího. Samotný model aplikace se aktualizuje z exportovaných XML souborů a deployment skriptů spuštěných v jednotlivých verzích balíčku.

## 5.2 Prototyp

Již v úvodu kapitoly jsem se zmínil, že je nutné ověřit návrh informačního systému. Vzhledem ke konceptu frameworku byla zvolena metoda pomocí prototypu. Prototypem ověřujeme především správné navržení datové vrstvy – máme uložena všechna potřebná data a vazby mezi jednotlivými objekty jsou realizovatelné a máme k dispozici všechna potřebná data.

Druhou částí ověření jsou komunikační rozhraní. Jedná se o ověření základní funkčnosti a návrhu rozhraní pro komunikaci s dalšími vnějšími systémy nutné pro plné fungování celého systému jako celku. Z komunikačních rozhraní se jedná především o následující:

- Napojení na technologickou část – odeslání zpráv.
- API pro technologickou část – změny stavů služeb.
- Odeslání záznamů do ticketovacího systému.
- Napojení na marketingový nástroj MailChimp.

Z pohledu procesů se jedná o ověření základních procesů nutných pro samotné fungování prototypu, které budou v dalších iteracích vývoje rozšířeny do finální podoby.

V jednotlivých částech kapitoly jsou blíže rozebrány jednotlivé moduly a jejich implementace v rámci prototypu.

### 5.2.1 Blok marketingových nástrojů

V rámci prototypu byly realizovány hlavní části modulu, které se týkají především definice služeb, samotné skladby produktů a balíčků. Dále byly realizovány kompletně ceníky a zadání jednotlivých cen k produktům. Při realizaci prototypu vyplynula z návrhu chyba – závislosti služeb byly definovány na dvou místech zároveň. Tento stav nedával smysl, bylo přistoupeno k ponechání závislostí na jiné služby pouze u definice produktu. Zároveň závislost

může být pouze na samostatných službách, závislost na doplňkových službách by nedávala smysl.

U definice ceníků bylo v datovém modelu zjištěno, že je vyžadována až příliš velká povinnost u datové položky cenového řádku. Byly zrušeny povinnosti na cizí klíče položky balíčku při definici cenového řádku, abychom umožnili vůbec zadání doplňkových služeb u balíčku.

Při realizaci části ceníků a definici produktů jsme také dospěli k části, kde nebyla nikde zmíněna zpětná vazba mezi službami a produkty/balíčky. Nikde nebyla ošetřena kontrola konzistence při změně definice elementární služeb, konkrétně změně atributu doplňková či samostatná. Do prototypu byly proto přidány kontroly, které tyto změny nedovolí provést, pokud bychom porušili integritu nějakého produktu. Uživateli se vždy zobrazí detailní chybové hlášení, kterého produktu a jeho položky se porušení týká.

V části pro propojení služeb byl navržen základní komunikační protokol pomocí SOAP zpráv. Bohužel v této fázi realizace nebylo v technologické části možné kompletně testovat komunikaci. Firma realizující implementaci technologické části nebyla s vývojem tak daleko. Jejich API pouze umělo měnit stav služeb v obchodní části podle první URL adresy přijaté ve zprávě. Nebylo možné simulovat všechny stavy.

Vzhledem k faktu, že míra nezávislosti mezi oběma systémy je velká, nebyl to problém pro další vývoj. Nastavení všech hodnot a stavů, které se odesílají do technologické části, je v rukou administrátora systému. Je pouze nutné nadefinovat název parametrů v technologické části a možné stavy dané kategorie služeb. To jsou jediné dvě styčné plochy. U stavů služeb byla jejich definice ještě rozšířena o adresu API, na které je možné je změnit. Jedná se o nutný požadavek pro generování zpráv odesílaných do technologické části, abychom nemuseli mít adresy napevno uložené v datové transformaci.

### 5.2.2 Blok objednávkového systému

V rámci prototypu byla implementována celá objednávka, včetně všech jejich detailů a návazností. Cílem bylo ověřit, zda je možné podle návrhu samotnou objednávku technicky složit na datové úrovni. Pro její úspěšné vyřízení je totiž nutné mít kromě obchodních položek i technické informace o jednotlivých objednaných produktech a jejich službách.

Při realizaci byly zjištěny drobné nedostatky v položkách ceníku – chybějící doba objednání produktu. Hodnota velice důležitá pro výpočet data možného užívání produktu a jeho expiraci. Tato položka byla do ceníků doplněna.

Další částí realizovanou v prototypu bylo ověření základního procesu objednávky – její příjetí a odeslání jednotlivých položek ke zpracování do technologické částí, včetně dodržení všech závislostí u jednotlivých položek. Celý základní proces je složen z kombinace zpracování pomocí front zpráv a stavových automatů. Položka je ve frontě zpráv vytvořena při změně stavu (toto lze přímo konfigurovat). Byly vytvořeny 3 základní fronty – fronta pro ob-

jednávky po odeslání, detaily řádku objednávky a ještě technická konfigurace řádku objednávky. Důvody pro tři základní fronty jsou následující:

- Každá fronta je pro jiný typ položky.
- Každá z vyjmenovaných položek obsahuje stavy, na které je potřeba při jejich změně reagovat.
- Fronty se ovlivňují hierarchicky od detailu detailu po samotnou objednávku.

Každá fronta má nastaven odpovídající příkaz pro danou položku, který je zpracován automaticky. Podle výsledku je pokračováno pak dále. Detaily detailu řádku (samotné služby, které se odesílají do technologické části) jsou měněny externě, pomocí komunikačního API z technologické části. Při změně položky jsou pak emitovány další změny, které mohou vystoupat až do samotné objednávky a výslednému stavu zřízeno.

Lze říci, že validace základního procesu objednávky proběhla na výbornou a v analýze a návrhu byly zpracovány všechny důležité části nutné pro budoucí provoz.

### 5.2.3 Blok komunikace

V této části prototypu bylo především nutné vyzkoušet a realizovat napojení na externí systémy pomocí jejich rozhraní.

U ticketovacího systému se jednalo pouze o zajištění možnosti vytvořit v systému ticket, který obsahuje odpovídající data. Toto propojení se podařilo realizovat. Bylo využito faktu, že jsme věděli, jaký ticketovací systém by měl být použít v produkčním prostředí. Tím se zjednodušilo testování a ladění komunikace, protože bylo možné využít vlastní instanci ticketovacího systému pro testovací účely.

Dále bylo v prototypu připraveno a otestováno obecné workflow pro založení ticketu na základě předaných vstupních dat. Workflow je možné využít v jakékoliv části budované aplikace napříč všemi moduly.

Při realizaci hromadné komunikace v prototypu byla situace o poznání složitější. Bylo nutné otestovat a odladit napojení na externí systém MailChimp přes REST API.

V rámci prototypu se podařilo realizovat samotné napojení na aplikaci MailChimp a realizaci základní komunikace. Samotné rozhraní je velice dobře popsáno, přesto skýtá své záludnosti při požadavcích na vytvoření nových položek, protože musí být přesně dodržena definovaná datová struktura zpráv.

V tomto bloku prototypu se oproti všem předchozím nevalidoval primárně datový model, ale napojení na externí systém. Napojení se podařilo úspěšně realizovat a bude pak rozvinuto do finální podoby v dalších iteracích při vývoji systému.

### 5.2.4 Blok správy zákazníků

V prototypu jsou obsaženy všechny hlavní části bloku. Jedná se především o základní evidenci zákazníků, kontaktních údajů a dalších komponent CRM systému. Dále bylo realizováno přiřazování ceníků. Zde datový model z návrhu plnil svoji funkčnost bez jakýchkoliv změn.

Druhou implementovanou částí je správa uživatelů a konfigurace rolí v rámci jednotlivých obchodních partnerů. Samotné uživatelské role byly rozšířeny o jednu položku – možnost, že jsou uživatelsky definovatelné. Toto bylo nutné z důvodu rozlišení základních interních rolí systému a rolí pro zákazníky. Systém byl konfigurován do dvou instancí rozdělených podle typu přihlašování. První instance plnila účel pro přihlášení interních zaměstnanců, druhá instance sloužila pro vyzkoušení funkčnosti klientské části.

### 5.2.5 Blok finančních operací

Z tohoto bloku bylo v prototypu realizováno úložiště dokumentů. Dále byla v prototypu ověřována základní funkčnost párování plateb k jednotlivým objednávkám. Poslední realizovanou částí je generování různých typů dokladů a jejich uložení v úložišti dokumentů.

V prototypu není vůbec realizována část komunikace se službou starající se o podepisování dokumentů ve formátu PDF. K této službě jsem neměl během realizace prototypu přístup. Rovněž jsem k ní neměl dodánu dokumentaci, aby bylo možné navrhnout a realizovat kompletní komunikaci.

Podobnou situaci potkal i platební modul. Tyto dvě komponenty nejsou v rané fázi pro systém důležité a jejich zapojení je možné kdykoliv v dalších iteracích následného vývoje po dodání všech specifikací komunikace.

Kapitola 6

# Testování, dokumentace

Každý softwarový produkt by měl správně projít během svého vývoje testováním a měl by být zdokumentován. Tyto části se prototypu samozřejmě také týkají. Obě části se vzhledem k použitému MDA frameworku realizují jiným způsobem než u klasického vývoje softwaru. Tento fakt vychází z podstaty, že aplikace je pouze modelována a nejedná se o klasickou implementaci zdrojového kódu.

## 6.1 Testování

Protože testování ve frameworku Origam je odlišné od klasického testování, zkusím nastínit jeho koncept. Při testování se především ověřují tyto části:

- Samotná funkčnost UI komponent.
- Základní práce s novým formulářem – založení, smazání a změna dat.
- Pokročilejší části – workflow, stavové automaty, transformace a různé typy validačních pravidel.

U testování základní funkčnosti formuláře se provádí testování na základě vložení referenčních testovacích dat vhodných pro aplikaci. Data jsou pak několikanásobně využívána k dalším testům. Jedná o velmi jednoduchý test UI a jeho základní práce s ním. Testy nejsou většinou nijak automatizovány, provází je analytik/programátor/tester ihned po implementaci.

U pokročilejších částí již dochází často k testování určitých procesů. V této situaci se využívá dokumentace aplikace (blíže popsáno v kapitole 6.2), která detailně popisuje testované části. Pro testování se využívá vestavěný editor pro XSLT transformace. Editor umožňuje definovat vlastní vstup a tím přímo otestovat správnou funkčnost implementované transformace. V tomto editoru je možné také realizovat testování různých validačních pravidel, která jsou použita na různých místech aplikace. Může se jednat o pravidla starající se

o předvyplnění dat, validaci hodnot po uložení a podmínky pro spuštění konkrétního kroku ve workflow. Všechny tyto operace provádí opět programátor.

Pro lepší testovaní a validaci výstupu framework obsahuje speciální konfigurovatelný trace pro jednotlivá workflow. Zde jsou zachyceny jednotlivé kroky daného workflow, jejich datový vstup, výstup a všechny parametry, které vstupují do transformací. Poslední úrovní je přímo logování vybraných částí aplikace. Tím mohou být pouze validační pravidla, SQL dotazy či ještě detailněji jednotlivé kroky spuštěného workflow.

Nad prototypem byly během realizace provedeny základní testy a testy základních případů užití.

## 6.2 Dokumentace

Každý softwarový produkt stojí a padá na kvalitě své dokumentace. Je to jedna z nejdůležitějších věcí hned po samotném produktu. Bez kvalitní dokumentace nejsme schopni aplikaci porozumět, ať už se to týká programátorské či uživatelské stránky aplikace.

Framework Origam má jednu velmi zásadní výhodu – umí se sám dokumentovat. Pro neznalé je určitě nutné vysvětlení této funkce. Framework je schopen na základě informací, které má uloženy v modelu a na základě doplněných dokumentačních informací u některých položek modelu vygenerovat kompletní online dokumentaci. Dokumentace obsahuje tyto části:

- Kompletní datový model – specifikaci všech entit v databázi, jejich název, typ, povinnost a popis.
- Kompletní přehled datových struktur a jejich složení.
- Dokumentaci celého menu aplikace.
- Diagramy všech stavových diagramů v aplikaci.
- Diagramy všech workflow realizovaných v aplikaci.
- Dokumentaci realizovaných transformací včetně ukázky příkladu užití.

Tato autodokumentace pokrývá především technickou dokumentaci aplikace. Uživatelská část dokumentace je řešena příručkou obsahují základní informace o ovládacím rozhraní a práci s ním. Dále se pro uživatele realizují školení pro práci s jednotlivými částmi aplikace.

Technickou online dokumentaci bohužel není úplně možné přenést do offline podoby. Jedná se o složitou kaskádu webových stránek. Tato dokumentace je dostupná všem uživatelům aplikace.

# Závěr

Na začátku tohoto projektu jsem si stanovil základní cíle, jimiž bylo provést analýzu, návrh a prototyp informačního systému postavený na základě předchozích výstupů, který slouží jako ověření návrhu.

Z počátku práce na samotné analýze informačního systému mírně vázla z důvodů nedostatků v dodané specifikaci. Tyto problémy byly řešeny pomocí osobních schůzek s odpovědnými osobami ze strany zadavatele systému. Souhrnná délka analytických schůzek by se mohla počítat na týdny. Tímto se podařilo doplnit chybějící dílky v dokumentu specifikující požadavky systému. V některých místech byla samotná specifikace systému doplněna velmi podrobně, v jiných místech pouze mírně doplněna či upravena. Na základě dostatečného množství informací bylo přistoupeno k samotnému návrhu informačního systému, jeho modulů a komunikačních rozhraní.

Na závěr práce byla provedena realizace prototypu pro ověření návrhu. V prototypu se podařilo úspěšně realizovat všechny základní komponenty. Návrh byl úspěšně validován a připraven na další iterace a pokračování ve vývoji. Můžu tedy říci, že stanové cíle byly úspěšně splněny.

Bohužel po dokončení prototypu a celé analytické části proběhly ve společnosti, pro kterou byl projekt realizován, velké personální změny a celý projekt nového informačního systému byl na neurčito odložen. S největší pravděpodobností k realizaci systému nikdy nedojde a zůstane celkově pouze „na papíře“.

Přestože byl projekt ukončen, nelze říci, že všechny části ze systému jsou dále nepoužitelné. Především, v prototypu byla nejnižší vrstva modelu realizována maximálně obecně, aby dané komponenty bylo možné využít i při jiné implementaci ve frameworku Origam. Z těchto obecných a znovu využitelných komponent zůstaly následující: obecná definice služeb jako prodejní položky, napojení na externí systém ticketingu, napojení na systém MailChimp a také napojení na další typ telefonní ústředny. Nad těmito obecnými moduly dále probíhá vývoj, aby je bylo možné v budoucnu nabídnout v rámci frameworku Origam jako předpřipravené řešení.

Komponenta pro napojení na telefonní ústřednu již byla dokonce nasazena

na jiném projektu do produkčního prostřední. Vývoj dalších komponent je v téměř finální fázi a budou v blízké době zařazeny do knihovny dostupných modulů.

Během samotné realizace prototypu byly také odhaleny drobné chyby v jádře frameworku Origam a bylo přispěno k jeho zlepšení a zvýšení stability.

# Literatura


[1] Object Management Group: Committed Companies And Their Products [online]. Únor 2014. Dostupné z: http://www.omg.org/mda/committed-products.htm

[2] Object Management Group: *Model Driven Architecture - A Technical Perspective*. První vydání, 2001, ormsc/2001-07-01. Dostupné z: http://www.omg.org/cgi-bin/doc?ormsc/01-07-01.pdf

[3] Object Management Group: *MDA Guide rev. 2.0*. Druhé vydání, 2014, ormsc/2014-06-01. Dostupné z: http://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf

[4] Sommerville, I.: *Software Engineering*. Pearson Education, 10 vydání, 2016, ISBN 978-1-292-09613-1.

[5] Pastor, O.; España, S.; Panach, J. I.; aj.: Model-Driven Development [online]. *Informatik-Spektrum*, ročník 31, č. 5, 2008: s. 394–407, ISSN 0170-6012, doi:10.1007/s00287-008-0275-8. Dostupné z: http://dx.doi.org/10.1007/s00287-008-0275-8

[6] Mieritz, L.: Gartner Survey Shows Why Projects Fail [online]. *Thisiswhatgoodlookslike*, Červen 2012, [cit. 2015-10-11]. Dostupné z: https://thisiswhatgoodlookslike.com/2012/06/10/gartner-survey-shows-why-projects-fail/

[7] Project Success Rates – Progress Over Time [online]. *Maturity and Tools Matter*, 1998 – 2016, [cit. 2015-10-11]. Dostupné z: http://www.unanet.com/content/project-success-rates-%E2%80%93-progress-over-time-maturity-and-tools-matter

[8] Rollings, M.: Why projects fail? Hint - It's not technical skills [online]. *Gartner Blog Network*, Březen 2013, [cit. 2015-10-11].

Dostupné z: http://blogs.gartner.com/mike-rollings/2013/03/28/why-projects-fail-hint-its-not-technical-skills/

[9] Whyte, A.: Gartner: 75% of all ERP projects fail - But why? [online]. *Office of Finance*, 2013 – 2014, [cit. 2015-10-11]. Dostupné z: http://officeoffinance.com/gartner-75-of-all-erp-projects-fail-but-why/

[10] Haan, J. D.: MDA, Model Driven Architecture, basic concepts [online]. *The Enterprise Architect*, Leden 2008, [cit. 2015-10-17]. Dostupné z: http://www.theenterprisearchitect.eu/blog/2008/01/16/mda-model-driven-architecture-basic-concepts/

[11] Haan, J. D.: MDE – Model Driven Engineering[online]. *The Enterprise Architect*, Leden 2009, [cit. 2015-10-20]. Dostupné z: http://www.theenterprisearchitect.eu/blog/2009/01/15/mde-model-driven-engineering-reference-guide/

[12] VanZandt, L.: Model-Driven Development [online]. *Object Management Group*, Leden 2008, [cit. 2015-10-14]. Dostupné z: http://www.omg.org/news/meetings/workshops/Real-time_WS_Final_Presentations_2008/Tutorials/00-T5_VanZandt-Mraidha_Part1.pdf

[13] MailChimp: Developer [online]. 2001 – 2015, [cit. 2016-2-20]. Dostupné z: http://developer.mailchimp.com/documentation/mailchimp/reference/overview/

[14] Lano, K.: *UML 2 sematics and applications*. Hoboken, NJ:Wiley, 2009, ISBN 978-047-0522-615.

[15] Pastor, O.; Molina, J. C.: *Model-driven architecture in practice*. New York: Springer, 2007, ISBN 978-354-0718-673.

[16] Microsoft Corporation: Microsoft SQL Server 2014 [software]. Duben 2015, požadavky na systém: Procesor x64, 1.4 GHz, 512 MB operační paměť, operační systém Microsoft Windows Server (2012, 2012 R2), Microsoft Windows (10, 8.1, 8. Dostupné z: https://www.microsoft.com/cs-cz/server-cloud/products/sql-server/Overview.aspx

[17] Microsoft Corporation: Microsoft Windows Server 2012 R2 Standard [software]. Duben 2012, požadavky na systém: Procesor x64, 1.4 GHz, 512 MB operační paměť, 32 GB diskového prostoru. Dostupné z: https://www.microsoft.com/en-us/server-cloud/products/windows-server-2012-r2/default.aspx

[18] Microsoft Corporation: Microsoft Internet Information Services 8.5 [software]. Září 2013, požadavky na systém: Procesor x64, 1.4 GHz, 512 MB

operační paměť, operační systém Windows Server 2012 R2. Dostupné z: http://www.iis.net/

[19] Visual Paradigm International Ltd : Visual Paradigm Community Edition 12 [software]. duben 2015, požadavky na systém: Intel Pentium, minimálně 2.0 GHz, 2 GB operační paměť, 4GB diskového prostoru, operační systém Microsoft Windows (XP/Vista/7/8,10), Microsoft Windows Server (2000/2003/2008/2012), Linux, Mac OS X 10.7.3 či vyšší. Dostupné z: https://www.visual-paradigm.com/

[20] Advantage Solutions, s.r.o: Origam 2015.11, aplikační server [software]. Listopad 2015, požadavky na systém: operační systém Windows Server (2008, 2008 R2, 2012, 2012 R2), prostředí .NET 4.6, IIS aplikační server, Microsoft SQL Server (2008, 2012, 2014). Dostupné z: https://origam.com/web/public/index

[21] Advantage Solutions, s.r.o: Origam 2015.11, vývojové prostředí [software]. Listopad 2015, požadavky na systém: operační systém Microsoft Windows Server (2008, 2008 R2, 2012, 2012 R2), Microsoft Windows (7, 8, 8.1, 10), prostředí .NET 4.6, Microsoft SQL Server, Origam aplikační server. Dostupné z: https://origam.com/web/public/index

[22] Anon: *Sběr požadavků na vytvoření informačního systému*. Praha, 2013.

[23] Anon: *Současný stav vybraných procesů komodizované divize*. Praha, 2014.

[24] Anon: *Rozdělení modulů systému na technologické a obchodní*. Praha, 2014.

[25] Vavrda, T.: *ORIGAM Platform*. Advantage Solutions s.r.o., Červen 2015.

Příloha A

# Seznam použitých zkratek

**MDA** Model driven architecture

**MDD** Model driven development

**MDE** Model driven engineering

**CIM** Computation independent model

**PIM** Platform independent model

**PSM** Platform specific model

**XML** Extensible markup language

**SQL** Structured query language

**API** Application programming interface

**REST** Representational state transfer

**MS** Microsoft®

**CSV** Comma-separated values

**PDF** Portable document format

**SMTP** Simple mail transfer protocol

**POP** Post office protocol

**IMAP** Internet message access protocol

**RSS** Rich site summary

**iCal** iCalendar

**IIS** Internet information services

**BI** Business intelligence

**UI** User interface

**XSLT** eXtensible stylesheet language transformations

**LDAP** Lightweight directory access protocol

**SOAP** Simple object access protocol

**HTTP** Hypertext transfer protocol

**GUID** Globally unique identifier

**GUI** Graphical user interface

**VOP** Všeobecné obchodní podmínky

**SMS** Short message service

**CRM** Customer relationship management

**SSO** Single sign-on

**ARES** Administrativní registr ekonomických subjektů

**URL** Uniform Resource Locator

**AD** Active Directory

**UC** Use Case

Příloha **B**

# Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `model` ..... adresář s exportovým modelem realizovaného prototypu
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `modely` ..... zdrojová data pro aplikaci Visual Paradigm
    - `pdf_tex` ..... zdrojová data exportovaných diagramů pro práci
  - `img` ..... ostatní zdrojová data obrázků pro práci
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF

Příloha C

# Seznam funkčních požadavků na informační systém

**8-1 M** Evidence uživatelských účtů

**8-2 M** Jiné formy autentizace

**8-3 M** Vytvoření uživatelského účtu zákazníka

**8-4 M** Vytvoření uživatelského účtu zaměstnance

**8-5 M** Úprava uživatelského účtu, změna hesla

**8-7 M** Deaktivace uživatelského účtu

**8-8 M** Blokace uživatelského účtu

**8-9 M** Přidání nebo odebrání formy autentizace uživatelského účtu

**8-11 M** Ověření e-mailové adresy

**9-1 M** Přihlášení uživatele v uživatelském rozhraní

**9-5 M** Odhlášení uživatele

**10-2 M** Nastavování práv uživatelů

**11-1 M** Evidence a správa fakturačního kontaktu

**11-2 M** Práva superadmina k fakturačnímu kontaktu

**12-1 C** Balíček

**12-2 C** Varianty složení balíčku

**12-4 M** Minimální požadavky na data spravovaná modulem definice produktů

**12-6 C** Produkt určený vybraným zákazníkům

**12-7 M** Možnost vazby mezi službami

**12-8 M** Validace vazeb produktů

**12-9 M** Validace k běžícím produktům

**12-10 C** Identifikace balíčku

**12-11 C** Přechod mezi variantami produktů

**13-1 M** Ceník

**13-2 M** Hlavní a vedlejší ceníky

**13-3 M** Slevy

**13-4 M** Postup ocenění

**13-5 M** Minimální požadavky na data spravovaná modulem ceníků a slev

**13-6 M** Prioritizace ceníků

**13-7 M** Prostupnost ceníků

**13-8 M** Přiřazení ceníku dle parametrů

**13-9 S** Dočasná sleva na produkt/službu

**13-10 M** Manuální ocenění zůstatkové ceny

**13-11 S** Automatické ocenění zůstatkové ceny

**13-12 M** Ocenění

**15-1 W** Informace o rozpracované objednávce

**15-2 S** Šablony zpráv pro hromadné oslovení zákazníků

**15-3 S** Seznam zákazníků k hromadnému oslovení

**15-4 S** Hromadné oslovení zákazníků

**15-5 M** Odhlášení klienta ze zasílacích seznamů

**15-6 M** Odesílání různými komunikačními kanály

**16-1 M** Zavedení ticketů

**16-2 M** Operace s tickety a s jejich stavy

**16-3 M** Skupiny ticketů

**16-4 M** Vazby ticketů a podpora operací

**16-5 M** Přijetí emailu od klienta

**17-1 M** Minimální požadavek na data evidovaná objednávkovým modulem

**17-2 M** Konfigurátor objednávky

**17-3 M** Funkční validace objednávky

**17-4 M** Shodnost uložených dat objednávky

**17-5 M** Přihlášení při objednávání

**17-6 M** Košík

**17-7 S** Uložení rozpracované objednávky

**17-8 M** Kontrola zaplacení objednávky

**17-9 M** Uložení objednávky do objednávkového modulu pomocí fronty

**17-10 C** Obecná objednávka

**17-11 M** Udělení souhlasu s obchodními podmínkami

**17-12 M** Synchronicita objednávky

**17-13 M** Vytvoření skupiny objednávek

**17-14 C** Vazba mezi objednávkami

**18-1 M** Plánovaný proces modulu služeb přiřazených zákazníkům

**18-2 M** Data uložená v modulu služeb přiřazených zákazníkům

**20-1 M** Výzva k platbě

**20-2 M** Proces daňových dokladů a záloh

**20-3 M** Dobropisy

**20-4 M** Funkce dobropisu

**20-5 S** Automatizovaný přenos dokladů do účetního softwaru

**20-7 W** Nastavení uložení přeplatků

**20-8 M** Úhrada objednávek volnými zálohami

**20-9 M** Kombinace úhrady zálohou, platbou a dárkem

**20-11 M** Párování plateb

**20-12 S** Úložiště dokladů

**20-13 M** Informace fakturačnímu modulu

**20-14 M** Informace klientům o provedené platbě

**20-15 C** Potvrzení platby

Příloha D

# Seznam případů užití

## D.1 Správa uživatelských účtů

| | |
|---|---|
| UC 10 | **Správa uživatelů** |
| | Uživateli jsou externí uživatelé a zaměstnanci. Každý uživatel má svůj uživatelský účet. O uživatelských účtech jsou evidovány informace typu jazyk, email, telefon atd. |
| UC 10-05 | **Založení a správa uživatelského účtu uživatele (uživatelem)** |
| 8-1 M<br>8-2 M<br>8-3 M<br>8-5 M<br>8-9 M | UC zahrnuje založení a správu externího uživatelského účtu. Probíhá prostřednictvím některého se zákaznických webů. Při založení lze využít služeb 3. stran (Moje ID, Facebook). Uživatel vstupuje do systému stále pod stejnou identitou, přestože využívá různé druhy přihlášení. |
| UC 10-10 | **Změna přihlašovacích a osobních údajů uživatele ve službách 3. stran** |
| 8-2 M | Při změně přihlašovacích a osobních údajů 3. stran je potřeba tuto změnu zachytit v systému – nemusí se jednat pouze o přihlašovací údaje, ale také o osobní informace o uživateli. |
| UC 10-15 | **Založení a správa uživatelského účtu (zaměstnancem)** |
| 8-3 M | Jedná se o založení a provedení změn u externího uživatelského účtu zaměstnancem přes technickou podporu či jiný komunikační kanál. |
| UC 10-20 | **Založení a správa uživatelského účtu zaměstnance** |
| 8-4 M | Případ zahrnuje založení a správu uživatelského účtu zaměstnance. Předpokládá se možné využití AD či LDAP služby. U zaměstnance nebude možnost využít jiné druhy přihlášení. |

| UC 10-25 | **Deaktivace uživatelského účtu** |
|---|---|
| 8-7 M | Na přání uživatele bude možné jeho účet deaktivovat. Deaktivaci může provést odpovědný pracovník uživatelské podpory. |
| UC 10-30 | **Zablokování uživatelského účtu** |
| 8-14 M<br>9-6 M | Zablokování uživatelského účtu po opakované neúspěšné autentizaci v systému (chybné heslo). Blokaci může také nastavit zaměstnanec. |
| UC 10-35 | **Odblokování uživatele** |
| 8-8 M | Odblokování uživatelského účtu oprávněným zaměstnancem uživatelské podpory. |
| UC 10-40 | **Ověření emailové adresy** |
| 8-11 M | Po registraci či změně uživatelského účtu je potřeba znovu ověřit emailovou adresu uživatele. Prověření proběhne zasláním kontrolního emailu se speciálním odkazem. Vykonání ověření emailové adresy bude moci provést i pracovník uživatelské podpory. Pokud nebude ověřena emailová adresa, nelze provádět objednávky a všechny aktuální objednávky budou pozastaveny |
| UC 10-45 | **Ověření telefonního čísla** |
| 8-16 M | Uživatel může vyvolat ověření svého telefonního čísla zasláním speciální SMS s autorizačním kódem, který je časově omezený. Při změně telefonního čísla bude systém opětovně vyžadovat jeho ověření. |

Tabulka D.1: Správa uživatelských účtu

## D.2 Správa zákazníků

| UC 15 | **Správa zákazníků (fakturačních kontaktů)** |
|---|---|
| | Uživatel vytvoří fakturační kontakt (v jakémkoliv okamžiku). Fakturační kontakt je samostatný a je napojený na uživatele. Mohou se předvyplnit jeho osobní údaje. Uživatel může mít pod svým účtem několik fakturačních kontaktů. |
| UC 15-05 | **Správa kategorií zákazníků (FK)** |
| | Jedná se o správu kategorií zákazníků. Jednomu zákazníku lze přiřadit více kategorií. Na kategorii může být v budoucnu navázána další funkcionalita. Například kategorie VIP může být při řešení požadavků upřednostňována. |

| UC 15-10 | **Správa zákazníků** |
|---|---|
| 11-1 M | Fakturační kontakt je vždy přiřazen pod nějaký uživatelský účet, který jej spravuje a má roli superadmin. Při založení fakturačního kontaktu se tento uživatel automaticky stává superadminem. Evidují se zde všechny základní údaje známe z CRM systémů. |
| UC 15-15 | **Přidání/odebrání práv k fakturačnímu kontaktu dalším uživatelům** |
| 11-2 M | Superadmin může explicitně dalším uživatelům přiřadit plná či dílčí práva k fakturačnímu kontaktu. Sám stále zůstává superadminem. |

Tabulka D.2: Správa zákazníků

## D.3 Produkty

| UC 20 | **Technické nastavení systému - produkty a služby** |
|---|---|
| UC 20-05 | **Vytvoření služby** |
| 12-4 M | Tento případ užití zahrnuje vytvoření a změnu služby. Cílem je podchytit všechny služby, které jsou k dispozici. Součástí definice služby je určení, zda je samostatná či doplňková. U služby jsou pak definovány její parametry v technologické části a možné stavy. |
| UC 20-10 | **Vytvoření technologických vazeb mezi službami** |
| 12-4 M<br>12-7 M | Při vytváření služeb je potřeba mít možnost nastavit jejich závislost – postup založení v technologické části. Dále také možnost, které služba je doplňková. |
| UC 20-20 | **Vytvoření produktové řady** |
| 12-8 M<br>12-9 M | Zde se jedná o definici produktové řady. Produktová řada sdružuje několik produktů a služeb obdobného charakteru. Produktová řada nese název, případně popis. Jejich využití v systému je pro rychlejší navigaci a marketingové vyhodnocení. |
| UC 20-25 | **Vytvoření produktu** |
| 12-4 M<br>12-7 M<br>12-8 M<br>12-9 M | Základní vytvoření produktu tak, aby splňoval všechny své náležitosti. |
| UC 20-26 | **Vytvoření typu produktu** |
| | Typ produktu obsahuje produkty z jedné produktové řady, které se liší svojí specifikací. |

| UC 20-30 | **Vytvoření balíčku** |
|---|---|
| 12-1 C<br>12-2 C<br>12-3 C<br>12-4 M | Případ užití pokrývající základní konfiguraci a nastavení balíčku, který se skládá z jednotlivých produktů a doplňkových služeb. Zároveň svým složením splňuje základní definované podmínky na specifikaci balíčku. |
| UC 20-25 | **Přiřazení balíčku/produktu konkrétnímu zákazníkovi** |
| 12-6 C | Produkt či balíček může být nabízen omezenému okruhu zákazníků. |
| UC 20-45 | **Doporučení - vazby mezi balíčky a produkty** |
| 14-6 M | Zde se jedná o možnosti up-sellingu a cross-sellingu, které lze pro každý produkt či balíček definovat. |

Tabulka D.3: Produkty

## D.4 Ceníky a slevy

| UC 30 | **Ocenění** |
|---|---|
| 13-1 M<br>13-6 M<br>13-7 M | Ceníky obsahují ceny balíčků a produktů po jejich jednotlivých položkách. Finální cena balíčku/produktu je zákazníkovi spočítána v objednávce po nastavení všech vlastností objednávané položky. Slevy jsou řešeny pomocí doplňkových ceníků. |
| UC 30-05 | **Vytvoření hlavních ceníků** |
| 13-2 M | Případ užití popisuje vytvoření a správu hlavních ceníků. Musí vždy existovat minimálně jeden hlavní ceník, který má přiřazen každý zákazník. Obsahuje všechny produktové položky. |
| UC 30-10 | **Vytvoření vedlejších ceníků** |
| 13-3 M | Doplňkové ceníky, které obsahují pouze vybrané položky. Mohou být přidělovány dynamicky. Zároveň plní funkci slev. |
| UC 30-15 | **Ocenění – přiřazení ceny** |
| 13-12 M | Pod tímto případem užití myslíme přiřazení výsledné ceny jednotlivým produktů a balíčkům. Z ceníků jsou vybrány ceny vázající se ke konkrétní položce. V případě, že nalezených cen je více, je vždy vybrána ta nejvýhodnější. |
| UC 30-20 | **Vytvoření ceníku oprávněných nákladů – pro zákazníky** |
| | Jedná se o ceník nákladů, které jsou odečítány od ceny zakoupeného produktu při jeho zrušení. |
| UC 30-21 | **Vytvoření ceníku nákladů** |
| | Ceník nákladů na realizaci jednotlivých služeb. |

| UC 30-25 | **Manuální ocenění zůstatkové ceny** |
|---|---|
| 13-10 M<br>13-11 M | Systém automaticky předpočítá zůstatkovou vratnou cenu pro zákazníka při zrušení produktu/balíčku. Tuto cenu bude muset schválit pracovník uživatelské podpory. Jedná se o poloautomatizovaný proces. |
| UC 30-30 | **Množstevní sleva** |
| | Možnost pro produkt/balíček definovat množstevní slevu při násobném nákupu. |

Tabulka D.4: Ceníky a slevy

## D.5 Objednávky

| UC 40 | **Objednávky** |
|---|---|
| 17-12 M | U položek objednávek existuje synchronní zpracování položky, ale také asynchronní. Na jakoukoliv fakturaci musí vzniknout objednávka. |
| UC 40-05 | **Přihlášení uživatele** |
| 17-5 M | Po přihlášení uživatele se ověří jeho služby, rozpracované objednávky a objednané služby, které nebyly ještě finálně vyřízené. |
| UC 40-10 | **Přiřazení ceníků závislého na faktoru** |
| 13-8 M | Uživateli je možné automaticky přiřadit ceník závislý na různých faktorech. Přiřazení ceníku může probíhat v různých fázích či periodicky. |
| UC 40-15 | **Uživatel hledá nový produkt/balíček** |
| | Uživateli se zobrazí všechny možné produkty/balíčky, které si může objednat. Zároveň se také nabízí cross-selling a up-selling k současným běžícím produktům. |
| UC 40-20 | **Uživatel potřebuje změnu/rozšíření stávající služby** |
| 17-2 M | Uživateli se zobrazí aktuální běžící balíčku/produkty. K těmto balíčkům se nabízí možnost změny. |
| UC 40-25 | **Uživatel potřebuje prodloužení balíčku/produktu** |
| 17-2 M | Uživateli se zobrazí aktuálně běžící produkty, kterým se blíží doba expirace. U jednotlivých položek se nabízí možnost prodloužení. |
| UC 40-30 | **Uživatel chce ukončit běžící produkt** |
| 17-2 M | Uživateli se zobrazí běžící balíčky a produkty. U jednotlivých položek se nabízí možnost ukončení za stanovených podmínek. |
| UC 40-31 | **Uživatel chce objednat správu služby** |
| 17-2 M | Pod správou služby se myslí objednání změny služby, atd. |

| UC 40-35 | **Uživatel vybírá do košíku** |
|---|---|
| 17-6 M | Uživatel podle předchozích případů užití vybírá jednotlivé položky do košíku. |
| UC 40-40 | **Uživatel se odhlašuje bez odeslání objednávky** |
| 17-7 S<br>3-6 C | Systém uloží aktuální stav košíku uživatele a nahraje jej při příštím přihlášení. |
| UC 40-41 | **Uložení neodeslané objednávky** |
| | Systém automaticky uloží rozpracovanou objednávku při odhlášení. Jedná o podobný případ užití jako UC 40-40. |
| UC 40-45 | **Uživatel se přihlásí a má uloženou objednávku** |
| 17-4 M | Po přihlášení se zobrazí rozpracovaná objednávka a obnoví se ceny jednotlivých položek objednávky. |
| UC 40-50 | **Uložení anonymních objednávek** |
| 3-6 C | Systém bude automaticky ukládat anonymní objednávky (pokud nedojde k přihlášení) kvůli možnosti analýzy a zlepšení marketingu. |
| UC 40-55 | **Uživatel odesílá objednávku** |
| 17-3 M | Po odeslání objednávky se validují její jednotlivé položky a pak se odesílá dále k dalšímu zpracování. |
| UC 40-60 | **Udělení souhlasu s VOP** |
| 17-11 M | Pokud některé produkty vyžadují odsouhlasení VOP, je nutné tento souhlas udělit. Eviduje se datum odsouhlasení VOP. |
| UC 40-70 | **Provedení platby** |
| | Objednávka, která je za nulovou cenu, je ihned odeslána k realizaci. Pokud má objednávka nenulovou cenu, je podle typu platby proveden další krok. |
| UC 40-75 | **Kontrola zaplacení objednávky** |
| 17-8 M | Systém prochází nezaplacené objednávky a automaticky odesílá notifikace. Při změně stavu objednávka na zaplaceno, je objednávka ihned poslána do realizace |
| UC 40-80 | **Notifikace zaplacení objednávky** |
| | Systém upozorňuje klienty na nezaplacené objednávky. Spojitost s UC 40-75. |
| UC 40-85 | **Smazání nezaplacené objednávky** |
| | Systém automaticky po uplynutí lhůty zruší nezaplacené objednávky a odešle zákazníkovi informaci o této situaci. |
| UC 40-90 | **Uložení objednávky do systému k realizaci** |
| 17-9 M | Po odeslání objednávky zákazníkem je objednávka uložena do fronty ke svému zpracování. |

Tabulka D.5: Objednávky

## D.6 Servisní modul objednávek a služeb

| | |
|---|---|
| UC 45 | **Servisní modul objednávek a služeb** |
| UC 45-05 | **Kontrola expirací služeb – notifikace** |
| 18-1 M | Dávková kontrola expirací služeb. Při zjištění blížícího se termínu expirace systém odešle zákazníkovi upozornění. |
| UC 45-10 | **Kontrola expirací služeb – automatické prodloužení** |
| 18-1 M | Dávková kontrola expirací služeb a jejich automatické prodloužení, pokud je nastaveno. |
| UC 45-15 | **Kontrola expirací služeb – ukončení** |
| 18-1 M | Dávková kontrola expirací služeb a jejich ukončení, pokud nemají nastaveno automatické prodloužení či je zákazník ručně neprodloužil. |
| UC 45-20 | **Report dostupných produktů/balíčků** |
| | Vytvoření reportu pro zákazníka s balíčky a produkty, které si aktuálně může objednat. |

Tabulka D.6: Servisní modul objednávek a služeb

## D.7 Individuální komunikace

| | |
|---|---|
| UC 50 | **Komunikace se zákazníky – individuální** |
| | Komunikace může probíhat přes různé komunikační kanály. Systém je zde propojen s ticketingem a poskytuje mu sady vybraných identifikátorů. |
| UC 50-05 | **Správa šablon pro individuální komunikaci** |
| | Vytvoření, úprava a smazání různých šablon pro individuální komunikaci. |
| UC 50-10 | **Zaslání výzvy k platbě, zálohového a daňové dokladu** |
| | Zaslání bude inicializováno z jiné části systému. Zde bude provedena pouze finální operace s odesláním emailu a založením ticketu. |
| UC 50-15 | **Oznámení o zřízení, změnách a ukončení služeb** |
| | Notifikace zákazníkovi, že došlo ke zřízení/zrušení/změně služby. |
| UC 50-20 | **Informace o objednávkách a jejich stavech** |
| | Průběžné informace pro zákazníka o stavu vyřizování objednávky. |
| UC 50-25 | **Individuální nabídky** |
| | Individuální nabídky zákazníkovi. Jejich charakter je spíše technického rázu. |
| UC 50-30 | **Informace o rozpracované objednávce (existenci neobjednaného košíku)** |
| 15-1 W | Upozornění na rozpracovanou objednávku |

| | |
|---|---|
| UC 50-40 | **Zaslání odkazu na změnu hesla** |
| UC 50-45 | **Oznámení o odstávce, výpadku služeb** |
| UC 50-50 | **Oznámení o vytvoření nového účtu** |

Tabulka D.7: Individuální komunikace

## D.8 Hromadná komunikace

| | |
|---|---|
| UC 55 | **Komunikace se zákazníky – hromadná** |
| | Systém bude napojen na externí systém pro hromadnou komunikaci. |
| UC 55-05 | **Správa šablon pro hromadnou komunikaci** |
| 15-2 S | Vytvoření, úprava a smazání marketingových šablon pro komunikaci. |
| UC 55-10 | **Vytvoření seznamu zákazníků k hromadnému oslovení** |
| 15-3 S | Bude možno vytvořit a uložit konkrétní seznam zákazníků nebo definici výběrového filtru, který lze opakovaně využít. |
| UC 55-15 | **Hromadné oslovení zákazníků** |
| 15-4 S | Hromadné oslovení zákazníků dle uloženého filtru vybraným komunikačním kanálem. |
| UC 55-20 | **Přihlášení klienta ke komunikačnímu kanálu** |
| 15-7 C | Klient si zvolí, jakým způsobem chce dostávat marketingové informace. U technických informací je vyžadován minimálně email. |
| UC 55-25 | **Odhlášení klienta ze zasílacího seznamu** |
| 15-5 M | Vyplývá ze zákona. Platí pouze pro obchodní, reps. marketingové informace. |
| UC 55-30 | **Vytvoření kampaně** |
| | Jedná se o spojení šablony a zasílacího seznamu za cílem hromadného oslovení zákazníků. |

Tabulka D.8: Hromadná komunikace

## D.9 Fakturace

| | |
|---|---|
| UC 60-05 | **Vystavení daňového dokladu (i zálohového)** |
| 20-2 M | |
| UC 60-10 | **Vystavení výzvy** |
| 20-1 M | |
| UC 60-15 | **Upomínání platby** |
| 20-10 S | |

| | |
|---|---|
| UC 60-20 | **Převod zálohy na daňový doklad** |
| UC 60-25 | **Vyžádání zaplacení dokladu skrze platební modul** |
| | Vyžádání platby na platebním modulu pomocí inkasa z platební karty. |
| UC 60-30 | **Párování plateb** |
| 20-11 M | |
| UC 60-35 | **Párování volné zálohy** |
| 20-8 M | |
| UC 60-40 | **Uvolnění volné zálohy** |
| 20-15 C | |
| UC 60-45 | **Přijetí neadresované platby a její identifikace** |
| UC 60-50 | **Vystavení dopropisu** |
| 20-3 M | |
| UC 60-55 | **Informování klienta o proběhlé platbě a jejím stavu** |
| 20-14 M | |
| UC 60-60 | **Přenos dokladů do účetního systému** |
| 20-5 S | Automatizovaný přenos všech dokladů do účetního systému. |

Tabulka D.9: Fakturace

## D.10 Platební modul

| | |
|---|---|
| UC 65-05 | **Vyžádání realizace platby** |
| UC 65-10 | **Předání fakturačnímu modulu** |

Tabulka D.10: Platební modul