FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Návrh CRM systému pro ISP
Student:	Bc. Lukáš Korel
Vedoucí:	Ing. Petra Pavlíčková, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Cílem diplomové práce je zanalyzovat a navrhnout CRM systém pro lokálního poskytovatele internetových služeb. Zároveň na základě analýzy a návrhu provést implementaci vybrané části systému.

1. Popište CRM systém.
2. Proveďte analýzu funkčních a nefunkčních požadavků správců sítě.3. Proveďte analýzu vhodných technologií pro systém.
4. Navrhněte architekturu a design systému.
5. Naimplementujte prototyp vybrané části systému.
5. Proveďte ekonomické vyhodnocení a navrhněte možnosti rozšíření.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 9. prosince 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Návrh CRM systému pro ISP

Bc. Lukáš Korel

Katedra softwarového inženýrství
Vedoucí práce: Ing. Petra Pavlíčková, Ph.D.

4. ledna 2020

Poděkování

Děkuji paní Ing. Petře Pavlíčkové, Ph.D., za odborné vedení práce a poskytnutí mnoha vědomostí z praxe během studia. Současně děkuji panu Mgr. Ing. Vítu Kleinovi, Ph.D., za poskytnutí rad do života, konzultací a motivaci a radu, že „Odklad je nepřítel úspěchu“. Děkuji také mé rodině a přátelům za podporu po celou dobu studia.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze, dne 4. ledna 2020 .....

Odkaz na tuto práci

Korel, Lukáš. *Návrh CRM systému pro ISP*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato diplomová práce se zabývá tvorbou informačního systému pro poskytovatele internetových služeb, který využívá bezdrátové technologie. Práce obsahuje analýzu požadavků správců sítě na systém. Na jejich základě jsou sestaveny případy užití systému a uživatelské role. Pro popsání objektů u poskytovatele je sestaven analytický doménový model. Na základě analýzy je sestaven návrh řešení pomocí informačního systému. V návrhu jsou zkoumány a vybrány vhodné technologie pro výsledný systém. Součástí návrhu je softwarová architektura systému pro snazší pochopení navržené struktury systému. Pro pochopení, jaká data budou ukládána, je vytvořen objektový datový model. Dále je navrženo aplikační programové rozhraní, které umožňuje napojení různých aplikací na systém. Pro snadné použití systému uživateli je vytvořen návrh, jak by uživatelské rozhraní systému mělo vypadat. Podle návrhu je naprogramována serverová část systému s využitím a různých externích aplikací a knihoven pro snazší implementaci systému. Součástí implementace je také zabezpečení vůči neoprávněnému přístupu k datům, jelikož systém bude přístupný přes internet. Část práce se také zabývá vyhodnocením systému. Zabývá se finančními náklady na vytvoření a údržbu výsledného systému. Jsou zde také popsány limity vytvořeného řešení a popsány možnosti dalšího rozvoje systému. V příloze jsou veškeré zdrojové soubory, které byly vytvořeny v rámci této práce. Hlavním přínosem této práce je návrh řešení pro řízení plateb za internetové služby a vytvoření základu projektu, který je připraven k implementaci uživatelské části systému a dalšímu rozšiřování.

Klíčová slova: Informační systém, analýza, návrh, implementace, evidence klientů, síťová zařízení, komunikační spoje, poskytovatel internetového připojení, RouterOS, MongoDB, JavaScript, NodeJS, TypeScript, Webpack

Abstract

This diploma thesis deals with the information system creation for Internet service providers who use wireless technologies. The thesis contains the analysis of network administrator's requests to this system. Based on the analysis the cases of application of the system and user's roles have been created. The analytical domain model is created for the object description at the provider. The proposal of the solution is based on the analysis of the information system. The proposal part contains the information about appropriate technologies that have been examined and selected for the final system. The proposed software architecture of the system is used to easier understanding the proposed system structure. The created object data model in the proposal part helps to understand which data will be stored. Furthermore, the application of the programming interface is designed to enable the connection of the various applications to the system. It contains user interface design that enables to use the system by users in easier way. According to the proposal the server part of the system, which uses different external applications and libraries, has been programmed so that it enables better system application. The implementation contains the security part that protects from unauthorized access to the system data, so far as the system will be accessible via the Internet. One part of this thesis contains the system evaluation. This part deals with the financial costs of the final system development and maintenance as well as the description of the system limitations and possibilities for the next expansions. The attachment of this thesis contains all source files that have been created during the thesis. The main benefit of this work is considered to be in the proposed solution to payment management for Internet services. The proposal is implemented in this thesis and by now it is prepared for user's part implementation and further expansion.

Keywords: Information system, analysis, design, implementation, customer records, network device, communication connection, internet service provider, RouterOS, MongoDB, JavaScript, NodeJS, TypeScript, Webpack

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době k dosažení úspěchu na trhu pomáhají informačních systémy. Zvyšují rychlost komunikace se zákazníky, čímž se zvyšuje kvalita poskytovaných služeb. S rostoucí spokojeností klientů se zvyšuje pravděpodobnost udržení svých zákazníků, což je potřeba zejména ve vysoce konkurenčním prostředí, jako je poskytování internetových služeb. S rozvojem informačních technologií rostou požadavky na kvalitu a dostupnost služeb. K urychlení operací v podniku se nasazují podnikové informační systémy, které mohou sloužit s k různým operacím s daty. Takovými operacemi mohou být například kontroly plateb, sledování anomálií a poskytování služeb bez nutnosti komunikace s fyzickou osobou. Takové informační systémy šetří náklady a umožňují expanzi podniků na další trhy.

Téma práce si autor zvolil z důvodu, že malí poskytovatelé internetového připojení často nemají ani základní informační systém pro jednoduchou správu dat o zákaznících, síťových zařízeních a platbách. Často mají smlouvy pouze v papírové formě a seznam aktuálně nasazených zařízeních v síti nemají. Takovými poskytovateli často jsou malé firmy v odlehlých oblastech, kam nejsou zavedeny optické spoje s možností připojení k internetu. Připojení klientů řeší pomocí bezdrátové technologie WiFi, která umožňuje vytvořit spoj bez nutnosti placení dalších poplatků a dostat internetové připojení ke koncovým zákazníkům v oblastech, kam se ekonomicky nevyplatí vytvořit kabelové internetové přípojky. V této práci je proto kladen důraz na vytvoření informačního systému, který bude finančně nenáročný, ušetří práci administrátorům malých sítí a umožní předávat klientům informace o jejich platbách za služby.

V práci autor vychází i ze svých zkušeností, které získal v prostředí malých poskytovatelů bezdrátového internetového připojení. Účastnil se zejména tvorby infrastruktury poskytovatele, správy klientských síťových zařízení a detekcí anomálií v síti. Jeho činnost u poskytovatelů spočívala také v poradenství při výběru hardwarových a softwarových řešení do sítě poskytovatele.

Hlavní náplní této práce je tvorba systému pro správu zákazníků a síťových

zařízení internetového poskytovatele ve formě webové aplikace. Součástí práce je teoretická část, která pojednává o technologiích a principech, které jsou použity v této práci. Dále obsahuje analytickou část, na jejímž základě je vytvořen návrh systému. Podle návrhu je vytvořen prototyp serverové části systému. Po implementaci je vytvořeno vyhodnocení nákladů na vývoj, nákladů na provoz a údržbu celého systému a možnosti jeho budoucího rozvoje včetně vyhodnocení limitů vybraného řešení.

Systém je určen pro nasazení v síti se síťovými zařízeními s operačním systémem RouterOS od společnosti MikroTik. Tento operační systém je často nasazován právě u malých poskytovatelů pro jeho jednoduchou správu a přijatelnou cenu.

Základem analytické části bude stanovení funkčních a nefunkčních požadavků správců sítě na systém. Na jejich základě budou vytvořeny modely případů užití a jejich scénáře. Součástí bude také popis současných společných technologií a principů u malých poskytovatelů. Dále se bude zabývat průzkumem existujících řešení, které alespoň z části vyhovují požadavkům správců sítě. V závěru analytické části bude vytvořen analytický doménový model s popisem vyskytujících se entit a jejich vztahů u poskytovatele.

Na základě analytické části bude vytvořen návrh systému. Jeho součástí bude výběr vhodných technologií pro tvorbu řešení. Pro pochopení struktury aplikace bude vytvořena softwarová architektura systému. Navržen bude také objektový datový model, z nějž bude možné vyčíst, jaké atributy o jakých datových typech budou ukládány v databázi systému a jak budou záznamy provázány. Jelikož systém bude disponovat aplikačním rozhraním, bude jeho definice navržena rovněž v této části. Protože systém bude určen pro ovládání uživatelem, bude návrh obsahovat uživatelské rozhraní s jeho popisem.

V implementační části bude probíhat samotná implementace podle návrhu, její součástí bude výběr vhodných knihoven a nástrojů pro vývoj řešení. V této práci bude vytvořena serverová část systému.

Ve vyhodnocení bude vytvořen hrubý odhad nákladů na tvorbu, provoz a údržbu systému. Veškeré odhady budou vycházet z očekávaných nákladů pro rok 2019. Součástí vyhodnocení bude popis limitů vytvořeného řešení. Jelikož systém bude rozšiřitelný, budou ve vyhodnocení popsány možnosti budoucího rozvoje.

Kapitola 1

Cíl práce

Cílem diplomové práce je vytvořit systém pro řízení vztahů se zákazníky pro lokálního poskytovatele internetových služeb. Součástí je zjištění potřeb správců sítě, se kterými jim může informační systém usnadnit práci a některé části zautomatizovat. Dále je nutné, aby aplikaci byl schopen ovládat jakýkoli uživatel běžně používající webový prohlížeč. Je kladen důraz na vytvoření řešení, které nevyžaduje zásah do konfigurace síťových prvků. Systém musí být rozšiřitelný a postavený na technologiích, které mají perspektivu a nebudou svazovat systém licencemi, které by komplikovaly používání systému. To vyžaduje průzkum vhodných technologií. Systém je určen klientům poskytovatele pro získání aktuálních informací. Pro správce sítě bude hlavním přínosem přehledně uspořádaná struktura informací o klientech a jejich síťových zařízeních. Tím se správcům usnadní rozvoj sítě do budoucna. Systém pro použití nesmí vyžadovat žádnou konfiguraci na straně klienta a musí být schopen běžet na zařízeních, která jsou dnes běžně využívaná. To vyžaduje aby práce obsahovala analýzu technologií, které takový požadavek splňují. Cílem je tak zanalyzovat aktuální potřeby poskytovatele internetových služeb. Na jejich základech v kombinaci s moderními technologiemi navrhnout architekturu umožňující tvorbu rozšiřitelného řešení. Součástí bude návrh uživatelského a aplikačního rozhraní. V implementaci je cílem provést vytvoření prototypu části systému. Pro ověření ekonomické smysluplnosti řešení bude součástí práce ekonomické vyhodnocení a návrh možností k rozšíření systému.

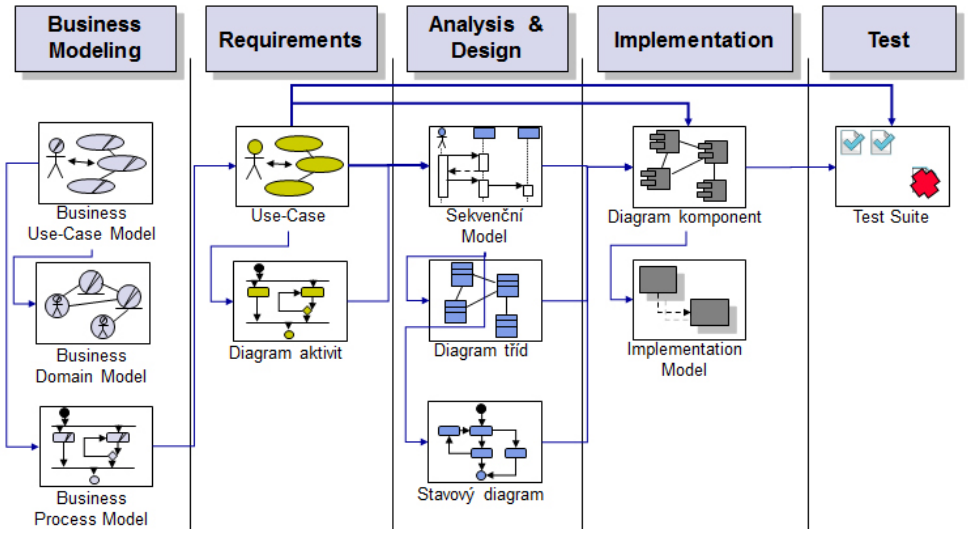
Kapitola 2

Teoretická část

V teoretické části jsou popsány základní principy a technologie využívané při vývoji softwaru. V práci je postupováno podle iterativního unifikovaného procesu vývoje.

2.1 UML

Unified Modeling Language (UML) [1, str. 28] je vytvořen pro vizuální modelování softwarových systémů. Dnes je aktuální verze UML 2.5.1 z prosince 2017. UML poskytuje mnoho modelů. Tato práce využívá pouze některé z nich.



Obrázek 2.1: Využití UML diagramů podle fází vývoje[2]

Na počátku se vytvoří diagramy případů užití (Use Case) pro grafické znázornění možností systému. Pokračuje se diagramem aktivit, který popisuje

jednotlivé kroky pro dosažení cílů v byznys procesu nebo v informačním systému. Pro tvorbu analytického doménového modelu se využívá diagram tříd (Class diagram).

Poté se projekt připraví na implementaci. Vychází se z již vytvořených modelů. Sekvenční diagram může modelovat případ užití. K popisu vazeb a datových typů základních stavebních prvků systému slouží diagram tříd. Aby bylo možné pochopit funkci systému, používá se stavový diagram. Znázorňuje, v jakých stavech se objekty nebo systém mohou nacházet a jaké jsou přechody mezi jednotlivými stavy. Před samotnou implementací se tvoří diagram komponent pro popis komunikace mezi většími celky systému.

2.1.1 Diagram užití

Pro znázornění možností systému se využívá UC diagram. Obsahuje takzvané aktéry, představující role lidí používajících systém. Každý aktér má různé možnosti použití systému. Například zákazník bude mít méně přístupných funkcí než správce aplikace. Jednotlivé případy užití mohou být propojeny s více aktéry. Každý případ má jednoznačný identifikátor a obsahuje krátký popis. Jednotlivé případy se poté textově upřesňují. [1, str. 93–101]

2.1.2 Analytický doménový model

Výsledkem analýzy je analytický doménový model. Slouží k identifikaci existujících objektů a modelují se pomocí analytických tříd. Třídy abstrahují objekty. Obsahují pouze klíčové atributy bez implementačních detailů. [1, str. 173–174] Mezi třídami jsou vytvořené relace. Relace popisuje vztahy mezi skutečnými objekty. U relace se eviduje její název. Často obsahuje i násobnost. Ta určuje počet objektů, které mohou být v libovolném okamžiku v relaci s objektem protější třídy. [1, str. 189–196]

2.1.3 Diagram aktivit

Diagramy aktivit jsou objektově orientované vývojové diagramy. Jsou modelovány pomocí uzlů propojených hranami. Ve specifikaci UML2 vycházení z principů Petriho sítí. Díky němu je možné používat různé typy cest. Tyto diagramy se obvykle připojují k ostatním diagramům pro upřesnění jejich chování. Používají se také pro modelování byznys procesů nebo detailů algoritmu. [1, str. 285–287]

2.1.4 Sekvenční diagram

Sekvenční diagramy jsou speciálním případem diagramů interakcí. Znázorňují uspořádání událostí mezi objekty. Pomocí nich se znázorňuje paralelní běh více objektů a interakce mezi nimi. Pro každý případ užití je možné vytvořit sekvenční diagram. [1, str. 253–259]

2.1.5 Třídní diagram

Diagram tříd obsahuje oproti analytickému modelu více implementačních detailů, aby z něj bylo možné vytvořit kostru implementace. Některé třídy z analytického modelu se mohou rozpadnout na mnoho dalších. Atributy jsou obohaceny o datové typy, třídy o metody a jejich parametry a viditelnost atributů a metod. Tento typ diagramu může obsahovat také abstraktní třídy a rozhraní tříd. [1, str. 340–343]

2.1.6 Diagram stavů

Každý objekt a systém se může nacházet v různých stavech. Stavový diagram modeluje stavový automat. Stavový automat modeluje dynamické chování objektu reagujícího na vnější události. Přechody mezi stavy mají směr a mohou se větvit. Přechody také mohou obsahovat časovou podmínku, při které dochází ke změně stavu. [1, str. 428–429]

2.1.7 Diagram komponent

Komponenta je část systému, která může být nahrazena jinou komponentou se stejným komunikačním rozhraním jako komponenta, kterou nahrazuje. Obsah komponenty je ostatním skrytý. Zveřejněno je pouze rozhraní, které komponenta vyžaduje nebo poskytuje. Komponenta samotná může v sobě ukrývat další komponenty. Kvalitní a promyšlená specifikace rozhraní je důležitá. Rozhraní po zveřejnění komponenty už nelze snadno měnit. [1, str. 391–395]

Speciálním případem komponenty je subsystém. Jedná se o komponentu popisující rozhraní celé vrstvy aplikace ve vícevrstvé architektuře. Například datová vrstva poskytuje rozhraní aplikační vrstvě a ta poskytuje rozhraní prezentační vrstvě.

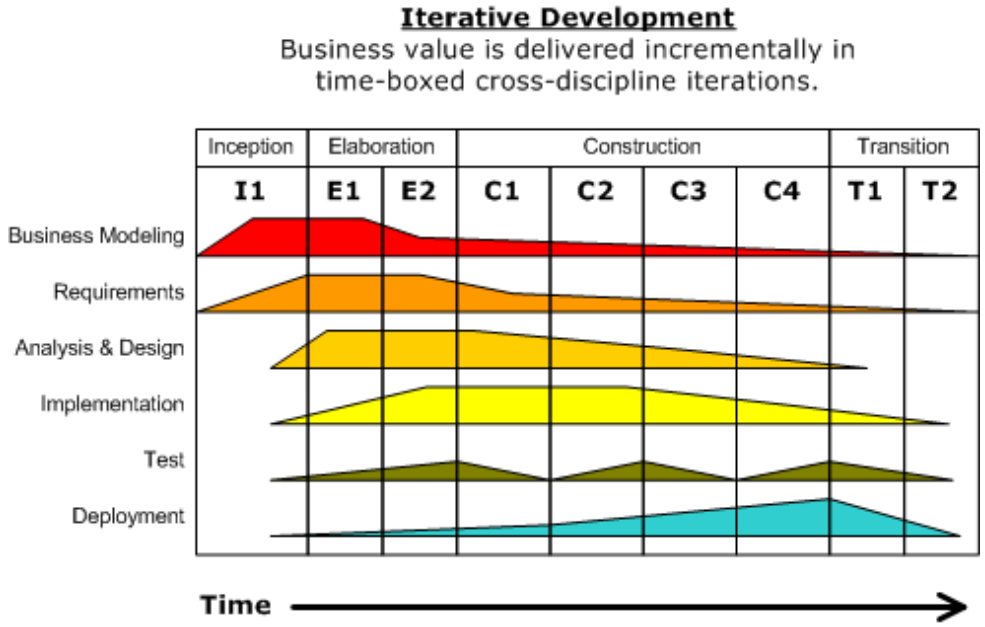
2.1.8 Diagram nasazení

Diagram nasazení spojuje komponenty, artefakty a uzly. Mapuje tedy architekturu softwaru na architekturu fyzickou. U distribuovaného systému se modeluje rozložení softwaru na fyzické uzly. Komponenta je typ softwaru, uzel je typ hardwaru a běhové prostředí. Mezi uzly jsou komunikační kanály. Uzel představuje prostředek pro nasazení a spouštění artefaktů. Artefakty jsou zdrojové a spustitelné soubory, skripty, databázové tabulky, dokumenty a výstupy vývojového procesu. [1, str. 468–472]

2.2 Životní cyklus softwaru

V dnešní době se pro tvorbu systému často využívá vývoj softwaru pomocí metodiky Unified Process (UP). Díky ní se zvyšuje pravděpodobnost kvalitně

dokončené softwaru. Určuje pracovníky, činnosti a produkty pro vytvoření funkčního softwarového systému. [1, str. 28]



Obrázek 2.2: Iterativní unifikovaný proces vývoje [3]

Na obrázku 2.2 je zobrazen jeden cyklus unifikovaného procesu softwarového vývoje rozdělený do čtyř fází vývoje. Příprava, rozpracování, tvorba a předání. Pracnost je rozdělena na byznys modelování, řízení požadavků, analýzu a návrh, vývoj, testování a nasazení. V této práci je do analýzy zahrnuto i byznys modelování a řízení požadavků.

2.2.1 Analýza

Slouží k definování požadavků na systém. Požadavky jsou základem pro plánování technického obsahu iterací, odhad nákladů a času na vytvoření systému. [4]

Analýza slouží k úplné specifikaci „Co bude tvořeno“. Pomáhá zákazníkovi, aby si ujasnil co potřebuje vyřešit a slouží softwarovým architektům, programátorům a testerům k vytvoření produktu ke spokojenosti zákazníka. Slouží také dodavateli k pochopení byznys procesů u zákazníka a vytvoření projektu na míru. Závěrem analýzy je analytický doménový model. [1, str. 137–138]

2.2.2 Návrh

Návrh nastává při přechodu z fáze rozpracování do fáze tvorby. Specifikují se v něm způsoby implementace požadovaných funkcionalit (jak bude tvořen

očekávaný produkt). Dochází k volbě vhodných programovacích jazyků, knihoven, databázových systémů a dalších prvků potřebných pro vývoj systému. Výsledkem má být návrh vycházející z analýzy, který lze implementovat. [1, str. 330]

V návrhu se také popisují softwarové komponenty a jejich vztahy. Popisuje se postup řešení problému. Je důležité dbát na nízkou provázanost jednotlivých komponent. Systémy se během života často upravují a změna v jedné komponentě nesmí způsobit nutnost úpravy jiné komponenty či nefunkčnost systému. [5, str. 169–171]

V návrhu se nejčastěji používají dva typy modelů. Strukturální a dynamické. Strukturální popisují statickou strukturu systému pomocí diagramu tříd, komponent a nasazení. Příliš detailní diagramy tříd není vhodné tvořit. O implementaci konkrétních tříd rozhodují programátoři. [5, str. 176–177]

Dynamické modely oproti statickým dokumentují dynamickou strukturu systému a znázorňují interakce mezi částmi systému. Dokumentují se sekvence požadavků na služby a změny stavů objektů pomocí sekvenčních a stavových diagramů. V návrhu postačují sekvenční modely pouze pro významné interakce. Ke každému UC však lze vytvořit sekvenční diagram. Podobné pravidlo platí pro stavové diagramy, ty se vytvářejí jen pro důležité a komplikované objekty. [5, str. 176–179]

Při návrhu je potřeba vytvořit specifikaci rozhraní. To je nutné dodržovat, aby bylo možné se systémem komunikovat. Pro usnadnění implementace se tvoří architektonický návrh. [1, str. 331–332]

2.2.2.1 Aplikační programové rozhraní

Informační systémy běžně mají aplikační programové rozhraní (API). Je specifikováno jako sada operací a poskytuje přístup k funkcím programu. Díky němu mohou funkce programu komunikovat s různými komponentami bez zásahu uživatele. [5, str. 660]

Každá vnitřní komponenta má své API. Databázové systémy poskytují rozhraní pro přístup k perzistentním datům. Knihovny poskytují rozhraní pro volání jejich funkcí, operační systémy pro volání systémových funkcí. K použití komponenty je potřeba pouze vědět, jak službu komponenty zavolat, není potřeba znát její vnitřní implementaci.

2.2.2.2 Uživatelské rozhraní

Při návrhu uživatelského rozhraní (UI) je definováno, jakým způsobem budou uživatelé systému používat jeho funkce a jak budou informace uživatelům zobrazeny. [5, str. 657]

Uživatelské rozhraní může být textové nebo grafické. V moderních informačních systémech je na GUI kladen velký důraz. Spokojenost uživatele se systémem závisí převážně na jednoduchosti jeho ovládání.

Z důvodu velkého množství různých velikostí zařízení a jejich displejů je potřeba dbát na správné zobrazení obsahu na každém zařízení. Tento princip se nazývá „Responzivní design“. Je sadou technik a technologií ke zpřístupnění aplikace co nejvíce uživatelům na nejrůznějších zařízeních. Pomocnými technikami jsou fluidní mřížky, flexibilní obrázky, dotazy na médium a další. Cílem je psát jeden kód spustitelný kdekoli. Tvorbou responzivního designu přispíváme k delší životnosti aplikace.[6, str. 16–18]

Jedna z technik tvorby responzivního designu je „Mobile First“. Ta vyžaduje vyvíjet UI postupně od nejmenších zařízení po největší. Základ stylů bude vycházet z mobilního vzhledu a pro větší obrazovky bude postupně upravováno. Tím se docílí toho, že většinu nastavení vzhledu nebudeme muset pro každý typ zařízení upravovat a pouze změníme některé vlastnosti pomocí speciálních dotazů na médium.[6, str. 29]

U webových aplikací je potřeba dbát na kompatibilitu nových funkcí se staršími prohlížeči. Je potřeba zjistit, jaké verze prohlížečů klienti používají a jim aplikaci přizpůsobit. Zajišťovat podporu a omezovat se starými technikami nemá smysl v případě, že verzi bez podpory nových technik používá pro nás zanedbatelné množství cílových uživatelů. Nemůžeme však využívat ty nejnovější funkce. Ty jsou podporovány pouze malým počtem uživatelských zařízení a zbytečně bychom omezili svou uživatelskou základnu. [6, str. 30]

2.2.3 Implementace

Implementace spočívá v převodu navrženého systému k reálnému spustitelnému výsledku. Vývojáři používají modely k pochopení požadavků a k rychlejší realizaci díky kvalitně vytvořenému návrhu. [1, str. 462]

Implementace je kritickou fází UP. Vzniká spustitelná aplikace. Zahrnuje vývoj v programovacích jazycích nebo přizpůsobení a adaptaci systémů. Aby vývoj nebyl extrémně časově náročný, některé části se používají opakovaně. Pro abstrakci problému se používají architektonické a návrhové vzory. Dále se používají knihovny či celé aplikační systémy. Při jejich použití je potřeba dbát na licenční podmínky. Často se pro vývoj používají open source knihovny a komponenty, čímž se šetří pořizovací náklady. [5, str. 183–184]

Implementace většinou probíhá na jiných platformách, než na jakých bude systém nasazen. Proto je potřeba již během implementace průběžně testovat na cílové platformě. Ta je pokud možno sestavena na straně dodavatele softwaru. Měla by odpovídat cílovému prostředí, to zobrazuje diagram nasazení. [5, str. 185–186]

Vývojáři často potřebují podpůrné nástroje. Hlavním nástrojem je IDE (Integrated Development Environment). To usnadňuje vývojářům vývoj v konkrétním programovacím jazyce. IDE poskytují funkce umožňující automatizovat některé postupy jako je překlad do strojového jazyka, kontrola syntaxe kódu, integrace komponent, automatické spouštění testů a mnoho dalších funkcí. [5, str. 186–187]

2.2.4 Testování

Testování slouží k prokázání funkčnosti softwaru a k nalezení chyb. Pro testování se vytváří testovací případy a data, nad kterými se testy provedou. Testovány by měly být všechny části a na každý požadavek by měl být minimálně jeden test. Testy se člení do dvou typů, testování vad a validační testování. Edsger Dijkstra poznamenal: *„Testy nemohou prokázat nepřítomnost chyb, mohou pouze prokázat jejich přítomnost“*. V testování se provádí také validace a verifikace. Barry Boehm definuje rozdíly takto: *„Validace ověřuje vývoj správného produktu, verifikace ověřuje správný vývoj produktu“*. Verifikace tedy ověřuje splnění funkčních a nefunkčních požadavků. Validace ověřuje splnění očekávání zákazníka. [5, str. 195–196]

Při testování se také provádí takzvaná inspekce. Jedná se o vizuální kontrolu kódu bez jeho spuštění. Kontrolují se softwarové modely, použité algoritmy, databázová schémata a další prvky. [5, str. 197]

Testování probíhá ve třech fázích. Při vývoji, před vydáním každé verze a u koncových uživatelů. Vývojové testování používá takzvané unit testování k otestování nejmenších částí softwaru. Testování komponent ověřuje rozhraní a interakce jednotlivých komponent. Systémové testování ověřuje systém jako celek. Čím dříve je chyba nalezena, tím je její odstranění levnější. Pokud je to možné, používají se automatizované testy, které se spouští při každé změně. Tím se šetří finanční prostředky a ověřuje se, zda drobná změna nezpůsobila nefunkčnost jiných funkcí. [5, str. 199–200]

Uživatelské testování je časově a finančně nejnáročnější typ testování. Ověřuje se jím robustnost systému a orientace uživatelů v systému. Je vhodné pracovat s koncovými uživateli již při vývoji. Takový uživatel by měl mít obecné znalosti byznys procesů u koncového zákazníka. Díky nim je možné důkladněji odladit systém k potřebám zákazníka. V další fázi se testuje v cílovém prostředí. Ověřuje se funkčnost systému v prostředí, kde se nachází další systémy. Je vhodné, aby zadavatel vyčlenil skupinu zaměstnanců, kteří budou uživatelsky testovat výsledný softwarový produkt před jeho převzetím. V závěru se provádí předávací testování. Zde se prokazuje funkčnost systému v cílovém prostředí, schopnost koncových uživatelů používat výsledný produkt a splnění všech požadavků. Po úspěšném předávacím testování je systém předán koncovému uživateli a dokončen hlavní vývojový proces. [5, str. 214–216]

2.2.5 Nasazení a údržba

Při dokončení produkční verze systému jeho vývoj nekončí. Krabicové verze softwaru si zákazníci nasazují sami, software na zakázku se nasazuje přímo do cílového prostředí. Oba typy softwaru potřebují během života podporu pro řešení případných problémů. Nasazení spočívá v konfiguraci přidružených systémů i produktu samotného.

Nasazený systém je potřeba dále rozvíjet, aby zůstal užitečný po dlouhé

období. Během evoluce se vytvářejí nové požadavky, kterým se musí systém během života přizpůsobit. Běžně se stává, že při užívání systému se nacházejí další chyby a ty je nutné opravit. Během života systému se mění hardwarová a softwarová výbava v produkčním prostředí. Proto je potřeba systém udržovat, aby byl schopný běhu i na moderních zařízeních.[5, str. 221–222]

V podnicích je životnost systému velmi dlouhá. V některých případech může dosahovat i několika desítek let. Podnikové systémy na míru jsou velmi nákladné a investici je potřeba rozložit do dlouholetého období. Aby bylo možné vyhovět vždy aktuálním požadavkům, vývojový cyklus ve opakuje. Sbírají se nové požadavky se kterými se tvoří analýza, poté se navrhují změny, které se následně implementují, testují a nasazují do produkce. U zákaznických systémů mohou software převzít interní vývojáři a údržbu s dodatečným rozvojem provozovat již bez externího dodavatele systému. [5, str. 222]

Údržba softwaru se dělí na opravu, přizpůsobení a rozšíření. Oprava vad zahrnuje opravu kódu, návrhu nebo chybné požadavky. Přizpůsobení prostředí nastává při změně v cílovém prostředí. Přidání nových funkcí nastává při změnách v byznysu koncového uživatele a tento typ údržby je nejnákladnější. V této fázi je obzvlášť oceňovaná kvalitní analýza a návrh systému. Pokud je systém navržen tak, že je snadno rozšiřitelný a komponenty nejsou silně provázané, náklady na údržbu jsou značně nižší. Součástí kvalitního softwaru musí být i kvalitní dokumentace. Ta při potřebných změnách usnadní pochopení systému novými vývojáři a zároveň z ní lze snáze odhadovat náklady na rozšíření softwaru. [5, str. 228–231]

2.3 Databázový systém

Databázový systém (DBS) umožňuje snadné udržení skutečností. Databázový systém se skládá ze systému řízení databáze (DBMS) a vlastní databáze (DB). [7, str. 20–29]

2.3.1 Database management system

Systém řízení báze dat (DBMS – Database management system) je počítačový program. Existuje řada produktů, komerčních i volně dostupných. Slouží k manipulaci s databází. Přijímá požadavky od uživatelů nebo aplikací přes rozhraní (například v jazyce SQL). Požadavky převádí na aktivity s databází a výsledek vrací v požadovaném formátu. Je to tedy program poskytující rozhraní mezi databází a aplikací či uživatelem. [7, str. 29–32]

Mezi aktivity DBMS patří tvorba a úprava databáze, datových a podpůrných struktur, čtení a změna dat, kontrola integrity dat, zabezpečení dat, ochrana před kolizí, autorizace, zálohování a obnova dat a další aktivity. Je také možné datům určovat omezení a pravidla neboli omezení referenční integrity. Obsahuje také řadu nástrojů pro vizualizaci struktury databáze a

dalších komponent. Na tento systém jsou kladeny velké nároky, jelikož udržuje v řadách případů velmi cenná data. [7, str. 29–32]

2.3.2 Databáze

Databáze je kolekce souvisejících záznamů obsahující vlastní popis. Související záznamy mohou být ukládány v různých formách (například v tabulkách). Vlastní popis obsahuje data o struktuře databáze, označují se jako metadata. V relačních databázích popisují názvy a vlastnosti tabulek, sloupců atd. Dále jsou zde obsaženy struktury pro zvýšení výkonu databáze. [7, str. 30–32]

Data v databázi se mohou nacházet v různých logických modelech. Mezi hlavní typy patří síťový, hierarchický, relační a objektový datový model. Volba modelu určuje prostředky pro vytváření struktury databáze (DDL) a prostředky pro tvorbu aplikací (DCL, DML, DQL a TCL). [8]

2.3.2.1 Relační model

Relační model databáze se vyznačuje jediným konstruktem a tím je relace. Relace má název, jména atributů v relaci a specifikaci domén atributů v relaci. Relace zároveň vytváří integritní omezení. Relační model obsahuje tabulky. Tabulka uchovává záznamy v řádcích a každý řádek obsahuje hodnoty. Pomocí selekce a projekce je možné provádět s daty sjednocení, průnik, rozdíl, kartézský součin a spojení. Tyto operace se provádějí pomocí jazyka SQL. [8]

K ukládání dat z objektových a objektově-orientovaných jazyků do relačních databází se často používá ORM. Objektové struktury se různými způsoby převádějí do relačního modelu. Některé objektové principy lze přenést snadno, některé komplikovaně. Mezi komplikovanější patří přenos dědičnosti objektů. Dědičnost lze řešit více způsoby. Jednou možností je nutné vytvoření jedné velké tabulky, která bude mít mnoho prázdných hodnot. Další možností je pomocí mnoha malých tabulek s mnoha vazbami. [8]

2.3.2.2 Objektový model

Do objektového modelu databáze je možné snadno přenést objektové principy. Objekty se skládají z dat a metod. Mezi objekty existuje skládání, dědění, závislost, klasifikace podle tříd, atd. Strukturované informace není třeba rozdělovat jako v relačním modelu. Protokol objektu je dán množinou přístupných zpráv, nikoli atributů. [8]

Jedna množina objektů může díky polymorfismu obsahovat objekty s různou strukturou dat i metod. Identita objektu je dána nejen vnitřními, ale i vnějšími vazbami. Klíče jsou interní záležitostí reprezentované objektovými ID (OID). [8]

Objekt je zde základním konstruktem. Je generován jako instance dané třídy, která nese informace o jménech atributů, specifikaci domén atributů, názvech metod atd. Stav objektu je specifický hodnotami atributů. Množinové

konstrukce jsou kolekce (set, bag, list, array, dictionary, atd.) a nad nimi se provádí množinové operace (so:, select:, collect:, detect:, inject:, reject:, intersect:, union:, atd.) pomocí jazyka OQL. [8]

2.3.2.3 Objektově-relační model

Každý přístup má své výhody a nevýhody, proto vznikl další model. Objektově-relační model je komerčně úspěšná kombinace relačního a objektového přístupu. [8]

2.4 Webová aplikace

Webové aplikace se od webových stránek liší tím, že uchovávají nějaký stav či se chovají různě podle uživatelského vstupu.

Webová aplikace je založená na principu klient/server. Na straně serveru běží služby, ke kterým se klienti připojují [5, str. 652]. V dnešní době je moderní způsob na straně serveru vytvořit rozhraní pro získávání sdílených dat a na straně klienta data přetransformovat do uživatelsky přívětivé podoby.

2.4.1 Hypertext Transfer Protocol

Jeden z nejpoužívanějších protokolů na aplikační vrstvě ISO/OSI modelu pro přenos dat k uživateli webové aplikace je Hypertext Transfer Protocol (HTTP). Protokol má 8 základní metod (GET pro získání obsahu, HEAD pro získání pouze hlavičky, POST pro odesílání dat na server, PUT pro odeslání změn na server, DELETE pro mazání obsahu na serveru, OPTIONS pro zjištění možností spojení se serverem, TRACE pro sledování dotazu poslaného na server a CONNECT pro navázaní TCP spojení skrze HTTP proxy, které se využívá pro HTTPS). Tento protokol funguje na principu požadavek/odpověď. Není tedy možné odeslat data ze serveru ke klientovi bez požadavku ze strany klienta. [9]

2.4.2 WebSocket

Protokol WebSocket (WS) řeší omezení HTTP, kdy není možné zaslat klientům zprávu bez jejich požadavku. WS funguje na principu, kdy se vytvoří plně duplexní komunikační kanál a server pak přes něj může zasílat data ke klientům. Dříve se často používal AJAX, kdy se periodicky klient ptá serveru na změnu dat, což vyžaduje větší datový tok z důvodu zbytečně častého opakování požadavků.

Protokol WS využívá při navázání komunikace HTTP, tím ale jeho závislost na HTTP končí. Jeho výhoda je že používá stejný TCP port jako HTTP (80 v otevřené podobě a 443 v případě šifrovaného spojení). Klient zašle požadavek na přepnutí z HTTP na WS. [10]

2.4.3 Representational State Transfer

Webové aplikace pracující s rozhraním pro základní datové operace používají často speciální typ API, takzvané REST (Representational State Transfer) s použitím HTTP. REST vychází z principu identifikovatelného prostředku s adresou URI (Uniform Resource Identifier). Veškerá interakce s prostředky probíhá pomocí metod POST, GET, PUT a DELETE. Metoda POST je určená k vytváření, GET čtení, PUT aktualizaci a DELETE odstranění záznamu. [5, str. 464, 660]

2.4.4 Simple Object Access Protocol

Simple Object Access Protocol (SOAP) je vhodné použít, pokud strana serveru poskytuje služby pro složitější výpočty a zpracování většího množství dat ve formě webových služeb. Aplikační logika je řešená na straně serveru. Posílaná data mezi a klientem a serverem jsou ve formátu XML. Jazyk popisující rozhraní služby je WSDL (Web Services Description Language). Popisuje názvy dostupných operací, typy parametrů a návratových hodnot, umístění a protokol služeb, port a URL. [11]

2.5 Podnikové informační systémy

V dnešní době nelze vytvořit univerzální IS pro celý podnik. Jelikož je každý podnik zaměřen pouze na nějaké odvětví trhu, má na systémy specifické požadavky. Některé podniky jsou orientované na výrobu, proto potřebují jiné systémy než podniky orientované například na prodej koncovým zákazníkům či podniky orientované na poskytování služeb. Na systémy je možné hledět podle toho, do jaké operační úrovně podniku patří. Jednotlivé úrovně popisuje globální architektura.

2.5.1 Globální architektura

Globální architektura rozděluje IS podle vertikální úrovně (členění podle práv a povinností zaměstnanců) a horizontální úrovně (členěné podle podnikových útvarů). Vertikální členění má 3 úrovně. Systémy transakčního zpracování, manažerské systémy a strategické systémy. V následujícím seznamu jsou popsány jednotlivé části globální architektury. [12]

TPS Transaction Processing Systems pořizují a aktualizují data, poskytují základní přehledy o podniku a poskytují data vyšším vrstvám. Neobsahují složité algoritmy, jsou co nejjednodušší, jelikož jejich hlavním úkolem je poskytnout rozhraní pro přístup k podnikovým datům a zároveň musí být spolehlivé. Do této kategorie patří CRM (Customer Relationship Management), RIS (Reservation IS), GIS (Geografic IS), MRP (Material

Resource Planning), ERP (Enterprise Resource Planning) atd. Tyto systémy pracují v reálném čase.

MIS Management Information Systems jsou systémy určené k řízení podniku na taktické úrovni. Jsou podobné i mezi podniky se zcela odlišným zaměřením cílového trhu. Jejich procesy patří do jedné ze tří linií: obchodně–logistické (nákup, prodej, sklady, ...), finančně–účetní (mzdy, pokladny, majetek, účetnictví, ...) a průřezové (správa organizace, marketing, legislativa, personalistika, ...). Do této kategorie patří systémy DSS (Decision-Support Systems) a BI (Business Intelligence), které mají za úkol pomoci při rozhodování, v manažerském plánování, pomoci s odhady a statistikami. BI často automatizovaným způsobem vyhodnocují data bez nutnosti ručního vyhodnocení. Poskytují automatizované reporty, statistické analýzy a algoritmy dataminingu.

ESS Executive Support Systems jsou systémy určené ke strategickému řízení podniku. Data získává z nižších vrstev i externích zdrojů, která agreguje, různě provazuje, vytváří trendy a prognózy. Zpracovávají data do hlubší minulosti a snaží se předpovídat vývoj do budoucna. Disponují pokročilými algoritmy pro analýzu dat, která mohou prezentovat i pomocí multimediálních prvků. Automaticky svá data a modely na nich postavené průběžně aktualizují v požadovaných intervalech. Využívají nástroje například s technologií OLAP (On Line Analytical Processing), jejíž základem je uložení dat v multidimenzionální databázi.

OIS Office IS je určen pro všechny vrstvy vertikálního členění systémů. Slouží jako podpora při kancelářské a týmové práci. Tyto systémy slouží k tvorbě, úpravě a přenosu dat administrativního charakteru v písemné i grafické podobě. Jeho hlavní výhoda je ve zvýšení produktivity administrativní činnosti. Nejznámějším příkladem je MS Office či LibreOffice.

EDI Electronic Data Interchange prochází všemi vrstvami vertikálního členění systémů. Tyto systémy poskytují nástroje pro výměnu strukturovaných dat pomocí dohodnutých standardů s mimopodnikovými subjekty. Propojují aplikace, převádí data do potřebných formátů a komunikují se systémy v cizích podnicích nejčastěji pomocí internetu.

2.5.2 Členění podle uplatnění

Podnikové informační systémy se dále dělí podle jejich uplatnění. Základní členění je do tří kategorií (řízení dodavatelských řetězců, plánování podnikových zdrojů a řízení vztahů se zákazníky). Ty mohou být v případě potřeby obohaceny o další systémy (systémy pro správu dokumentů, datové sklady, a mnoho dalších). V následujícím seznamu jsou popsány tři základní kategorie IS. [13, str. 66–90]

ERP Enterprise Resource Planning tvoří jádro v podniku. Zahrnuje aplikace pro řízení podnikových dat a automatizuje podnikové procesy. Často je rozšířen o další IS, se kterými interaguje. Součásti ERP se liší podle toho, o jaký typ podniku se jedná. Častou součástí ERP systému jsou aplikace pro podporu plánování projektů, výrobních zakázek, výrobních plánů, správu financí a řízení vnitropodnikových transakcí.

SCM Supply Chain Management je soubor nástrojů a procesů sloužící k řízení celého dodavatelského řetězce. Pomáhá zkracovat čas zpracování zakázek a zvyšovat spolehlivost dodání produktů. Jeho hlavní účel je sledování toků od materiálů po produkty. Jsou propojeny s dodavateli a odběrateli, se kterými vyměňují informace. Tyto systémy se skládají z komponent pro plánování, nákup, výrobu, expedici a reklamaci. Podporují plynulost v zásobování, zpřehledňují skladové zásoby a umožňují snížit nutné zásoby na minimum.

CRM Customer Relationship Management slouží k řízení vztahů mezi podnikem a zákazníky. Pomáhají při prodeji produktů a služeb a podporují udržení stávajících zákazníků i získání nových. Jejich účelem je vytvořit obousměrný komunikační kanál mezi podnikovým subjektem a jeho zákazníky. Tyto systémy obsahují aplikace a prostředky pro oblast podpory obchodních činností (prodej, marketing a podpora) a zákaznických služeb. Pomocí těchto systémů lze měřit klíčové ukazatele výkonnosti neboli KPI (Key Performance Indicator) a usnadňují zaměření reklamních akcí.

2.6 Bezpečnost informačních systémů

Bezpečností informačních systémů je potřeba se zabývat z více důvodů. Hlavním důvodem je, že se v datech může nacházet kapitál firmy, know-how atd. Jedním přístupem ochrana dat je před jejich neúmyslnou či ztrátou. Druhým přístupem je ochrana dat proti neoprávněnému přístupu, proti odcizení dat a proti úmyslnému zničení dat ze strany útočníka. [13, str. 122]

K jednotlivým systémovým funkcionalitám a datům mohou mít přístup jen oprávnění uživatelé. Tento požadavek je běžně řešen pomocí přístupových práv uživatelů, šifrovacích a hashovacích algoritmů pro ochranu přenášených a uložených dat, ochranu hesel, nepopiratelnost původu atd. K ověření uživatele dochází pomocí uživatelského jména a hesla, případně ověřovací SMS zprávy, různých klíčů atd. V současné době je velkým tématem GDPR a proto je nutné zajistit, aby k citlivým datům o fyzických osobách měli přístup pouze oprávnění uživatelé systému. Systémy je potřeba chránit nejen softwarově ale také fyzicky, aby nebylo možné se k datům dostat mechanickým způsobem (například zcizením datového nosiče s daty). [13, str. 122–123]

V následujícím seznamu je popsáno, čím se nauka o šifrování (kryptografie) zabývá a je nutné se tím v bezpečnosti informačních systémů zabývat také. [14]

Důvěrnost/Confidentiality Poskytuje ochranu před odhalením obsahu dat neoprávněným stranám. K tomu se v sítích využívá navázání zabezpečeného spojení.

Celistvost/Integrity Ochrana dat před neoprávněnou modifikací. Pokud dojde k modifikaci, je potřeba ji detekovat. Z dat musí být jasné, že byla pozměněna neoprávněným způsobem.

Autentizace/Authentication Identifikace, zda je subjekt skutečně ten, za který se vydává. Jde o jednoznačné určení identity. Subjektem mohou být fyzická zařízení, logické procesy a lidské subjekty. Autentizace je tedy ověření vydávané identity subjektu (probíhá na základě znalosti – heslo, vlastnictví – klíče od bytu/identifikační karta, charakteristické vlastnosti – biometrické informace).

Autorizace/Authorization Řízení přístupu, zajišťující ochranu před nepovoleným použitím prostředků. Někdy se do něj zahrnuje prokázání původu dat (autorství).

Nepopiratelnost/Non-repudiation U některých dat je potřeba zajistit nepopiratelnost původu. Slouží k prokázání provedení nějaké akce a zabránění jejímu pozdějšímu popření (např. bankovní transakci).

Data je nutné chránit i před ztrátou. Z toho důvodu se periodicky provádí zálohy dat. V některých případech nařizuje legislativa, aby bylo možné některé transakce doložit i po deseti letech, proto je nutné mít data kvalitně archivovaná. [13, str. 123]

2.6.1 Hashovací algoritmy

Hashovací algoritmy jsou z matematického pohledu jednosměrné funkce. Je snadné spočítat hash vstupu, ale nesmírně obtížné nalézt vstup pro konkrétní hash. Každá hashovací funkce není injektivní (není prostá), takže může nastat případ, kdy pro různé vstupy vznikne stejný hash (kolize). Je požadováno, aby systematické nalezení kolize bylo výpočetně velmi náročné. Nesmí existovat korelace mezi vstupními a výstupními bity. Funkce mohou mít na vstupu libovolně dlouhý řetězec, který převádějí na řetězec konstantní délky. Snahou je, aby kolize nenastala při podobných nebo mírně pozměněných vstupních datech, čímž hashe umožňují detekovat změnu obsahu ve zprávách. Tím se vytvoří otisk vstupního řetězce, který se nazývá hash a závisí na všech bitech vstupu. Hashe slouží ke kontrole integrity dat, při porovnání zpráv, vyhledávání,

indexování a vyhledávání v tabulkách pomocí oblastí se stejným hashem i tvorbu digitálních podpisů. [15, Hashovací funkce]

Dnes jsou v této oblasti nejpoužívanější algoritmy SHA. Jedná se o skupinu kryptografických hashovacích funcí (SHA-1, SHA-224, SHA-256, SHA-384 a SHA-512). SHA-1 má výsledný hash délky 160 bitů, zbylé verze (označovány jako SHA-2) mají délku hashe odpovídající jejich číslu. SHA se využívá v protokolech TLS, SSL, SSH, IPsec atd. [15, Hashovací funkce]

V ideálním případě by pro hashovací funkci platilo, že nelze nalézt vzorovou zprávu k vybranému hashi. Pomocí hrubé síly toho však docílit lze. Pro hash délky n lze hrubou silou nalézt vzorovou zprávu s 2^n výpočty a pro dvě rozdílné zprávy lze nalézt stejný hash s $2^{n/2}$ výpočty. To je založeno na jevu nazvaném „Narozeninový paradox“, který říká: „Ve skupině 23 náhodně vybraných lidí existuje 50% šance, že dva z nich budou mít narozeniny ve stejný den.“. Pro skupinu 57 lidí je taková šance již 99%. Takový pár se označuje jako kolize. Síla hashovací funkce je tedy polovina délky jejího výsledného hashe, proto je potřeba tvořit dostatečně dlouhé hashe. [15, Hashovací funkce]

Hesla do systémů nesmí být uložena v čitelné formě. Aplikace jsou velmi často zranitelné a útočníci se mohou k heslům uživatelů dostat. Šifrování hesel není dobrou volbou, jelikož k tomu musí být někde uložen šifrovací klíč. Pokud se tohoto klíče někdo zmocní, dokáže hesla získat. Proto se používá hashování. Samotné hashování hesel nestačí. Pokud nastane případ, že mají uživatelé stejná hesla, je i hash stejný. Pokud útočník zjistí, který řetězec vytvoří daný hash, jsou uživatelé se stejným hashem také zranitelní. Proto se přidává k heslu sůl (salt), tou musí být dostatečně náhodný řetězec znaků pro každého uživatele. Tím se odstraní problém stejných hashů se stejným heslem díky náhodnosti soli. Aby se útočníkům ztížilo hledání řetězcú odpovídající danému hashi, opakuje se hashovací algoritmus několikrát, což může být opět pro každého uživatele jiné a tím se ještě více zkomplikuje útok hrubou silou. [16]

V databázi se ke každému uživateli ukládá uživatelské jméno, použitá hashovací funkce (dostatečně odolná), počet cyklů hashování, náhodná sůl a výsledný hash. Po zadání přihlašovacích údajů uživatele systém vyhledá sůl, přidá ji k heslu, vypočítá hash pomocí hashovací funkce v cyklu tolikrát, kolikrát je uloženo v databázi a získá výsledný hash. Pokud se výsledek shoduje s hashem v databázi, je uživatelské heslo správné. K ochraně je nutné, aby měli uživatelé dostatečně složité heslo, aby nebylo možné se do systému dostat například pomocí slovníkových útoků. [16]

2.6.2 Šifrovací algoritmy

Na obrázku 2.3 je znázorněn obecný šifrovací a dešifrovací proces přenosu důvěrné zprávy. Na levé straně je aplikován šifrovací algoritmus s využitím jednoho klíče na důvěrná data. Vzniknou data zašifrovaná a ta se přenesou k druhé straně po nezabezpečeném komunikačním kanále. Na pravé straně

Obrázek 2.3: Obecný šifrovací proces [15, Úvod do kryptologie]

se zašifrovaná data dešifrují pomocí druhého klíče a získají se původní data. V případě, že jsou klíče stejné, jedná se o symetrickou kryptografii a dešifrovací algoritmus je prostou inverzí k šifrovacímu algoritmu. V případě, že jsou klíče různé, jedná se o asymetrickou kryptografii. V takovém případě se jmenují veřejný a privátní klíč. Tyto klíče však mají matematický vztah. Šifrovací a dešifrovací algoritmus je pak značně odlišný. [15, Úvod do kryptologie]

V kryptografii platí Kerckhoffsův princip podle Auguste Kerckhoffs von Nieuwenhoff z roku 1883. *„Bezpečnost šifrovacího systému nesmí záviset na utajení algoritmu, ale pouze na utajení klíče.“*

2.6.2.1 Symetrické šifrování

Symetrické šifrování používá pro šifrování i dešifrování stejný klíč. Symetrické šifry často používají bloky, do kterých jsou data určená k šifrování rozdělena. Takové šifry se nazývají blokové. Nejrozšířenější blokovou šifrou je AES a byla schválená v roce 2001 jako náhrada staršího typu. Délka klíče u AES může být 128, 192 nebo 256 bitů. [15, Symetrická kryptografie]

2.6.2.2 Asymetrické šifrování

Problém symetrického šifrování je ve výměně šifrovacích klíčů bez možnosti odposlechu třetí stranou. Asymetrická kryptografie tento problém řeší pomocí skupiny kryptografických metod, ve kterých se pro šifrování a dešifrování používají dva odlišné klíče (veřejný a soukromý klíč). Základem jsou jednosměrné funkce, které jsou obtížně invertovatelné v požadovaném čase na současných technologiích a současných znalostech. Pomocí veřejného klíče lze zprávu zašifrovat, nikoli dešifrovat, z toho důvodu může být volně zaslán bez utajení, protože je pro potenciálního útočníka k dešifrování zprávy nepoužitelný. Pro dešifrování je potřeba soukromý klíč, který musí zůstat v utajení. Tento princip se používá i pro podepisování zpráv (ověření původu). [15, Asymetrická kryptografie]

Moderní způsob v asymetrické kryptografii je použití ECC (šifrování na eliptických křivkách), které je založeno na algebraických strukturách eliptických křivek nad konečnými tělesy. Bezpečnost stojí na předpokladu, že nalezení

diskrétního logaritmu náhodného bodu eliptické křivky s ohledem na známý základní bod je za použití současných technologií a znalostí časově velmi náročné. Velikost eliptické křivky určuje složitost problému. Oproti staršímu RSA stačí ECC kratší klíče při zachování stejné náročnosti na prolomení (např. 256b ECC klíč je ekvivalentní 3072b RSA klíči). [17]

Asymetrické šifry se využívají i pro elektronické podpisy a certifikáty. Součástí hierarchie s jsou certifikační autority, které vydávají jimi certifikáty. Uživatel pak může pomocí veřejného klíče této autority ověřit, že certifikát nebo podpis, kterým byla data podepsána, je vydaný právě touto autoritou. Pomocí tohoto principu lze zajistit nepopiratelnost původu u podepsaných dat elektronickým podpisem.

2.6.2.3 Hybridní šifrování

Asymetrická kryptografie je oproti symetrické značně pomalejší. Proto se využívá pro šifrování přenášených dat symetrický algoritmus. Symetrický klíč se zašifruje asymetricky a odešle po veřejném kanále spolu s daty zašifrovanými symetrickým klíčem. Tento způsob se označuje jako hybridní šifrování. [15, Asymetrická kryptografie]

2.7 Softwarové licence

Software je nehmotná věc. Vztahuje se na něj autorský zákon a může být na něj udělena licence, podle které může být software užíván. Licence může být časově neomezená i omezená. Licence se uděluje ve formě licenční smlouvy (aktuálně podle občanského zákoníku). Licence stanovuje, zda a jakým způsobem je možné software měnit, spojovat s jiným softwarem, rozmnožovat, atd. Koncovým zákazníkem může být fyzická i právnická osoba. [18, str. 269–270]

Licence se dělí na výhradní a nevýhradní. Pokud je poskytnuta výhradní licence, nesmí být software licencovaný nikomu dalšímu. Používá se pro software vyvíjený na míru zákazníka. Licence nevýhradní neomezuje poskytovatele softwaru v dalším nakládáním. Ta se používá při vývoji softwaru určenému větší skupině zákazníků. [18, str. 270]

Dalším způsobem dělení licence je podle omezení rozsahu. Omezení podle územního rozsahu není v softwarových licencích často používáno. Omezení časového rozsahu se používá u licencí s předpokladem periodických platem za licenci a softwarové služby. Pokud časový rozsah není omezen, je licence trvalá. Při omezení množstevního rozsahu se často využívá multilicencí. U multilicencí jsou běžné množstevní limity. Omezení může být dáno počtem uživatelských stanic na který může být software instalován, celkovým počtem uživatelů softwaru nebo počtem učivatelů užívajících software v jeden okamžik. Další omezení mohou být dohodnuta ve smlouvě. [18, str. 271–273]

Komplikovanější je licencování produktů, které jsou tvořeny vlastním i cizím softwarem. U spojeného autorského díla může dojít ke střetu li-

cenčních podmínek. Proto je nutné zajistit, aby konfliktní software nebyl svázán s dodávanou aplikací. Řešením střetu je dodání softwarů s různou licencí, které společně pouze komunikují přes rozhraní, ale nejsou spolu svázány do jednoho díla. [18, str. 275–276]

Licenční smlouva musí obsahovat jednoznačnou identifikaci softwaru, který je jejím předmětem. Dále právo a způsob užití softwaru, rozsah licence, může obsahovat i licenční poplatky a platební podmínky a odměny z výnosu. Některé licenční smlouvy obsahují výhradnost, nárok na aktualizaci a opravu softwaru, oprávnění k pořizování rozmnoženin, podmínky pro zánik licence, omezení zásahu do díla, povinnost mlčenlivosti, možnosti převodu licence, povinnost aktivace softwaru, odpovědnost za vady produktu a mnoho dalších součástí. [18, str. 277–280]

Podle typu licence se software dělí do různých kategorií. Hlavní rozdělení je podle dostupnosti zdrojového kódu. Kategorie free software, opensource a public domain jsou kategorie nekomerčního softwaru. Freeware je šířen bez zdrojového kódu a tudíž jej nelze snadno upravovat. Některé typy softwaru jsou zdarma pouze pro nekomerční použití. Vývojáři adware jsou placeni z reklam, které jsou tohoto softwaru součástí. Dále existuje mnoho variant softwarových licencí pro různé způsoby použití. [18, str. 288]

2.7.1 Free a open source sofware

Tato kategorie se označuje jako FOSS. Free software klade důraz na svobodné užití, ale není zveřejněn jeho zdrojový kód. V některých případech jeho licence omezuje právo užití softwaru ke komerčním účelům, nebo je komerční užití zpoplatněno. [18, str. 289]

Open source software (OSS) je opakem proprietárního softwaru. Takový software má nejčastěji následující vlastnosti. OSS je volně šiřitelný, má volně dostupný zdrojový kód. Lze z něj tvořit odvozený software, pokud i ten bude vydán pod stejnou licencí. V licenci je možné vynutit si, aby odvozený software nesl jiný název, aby nemohlo dojít k jejich záměně.Lze jej používat i ke komerčním účelům. Pokud je produkt jednou OSS, nelze z něj již vytvořit proprietární, čímž se od free softwaru také liší. Detailnější informace jsou na `https://opensource.org/`. [18, str. 290]

Výhoda open source spočívá zejména v možnosti vývoje a údržby aplikace i po ukončení podpory ze strany dodavatele. Je to z toho důvodu, že základním principem takového softwaru je volná dostupnost zdrojového kódu. Neznamená to však, že se s takovým kódem může dělat cokoli. [5, str. 188–189]

Při použití open source kódu v aplikaci je vhodné sestavit procesy pro evidenci použitých komponent a jejich licencí platných v době jejich použití. Je potřeba znát práva a povinnosti k použití každé licence. U velkých projektů je vhodné zřídit systémy auditu softwarových licencí. [5, str. 189]

Pokud firma používá open source software, je vhodné ji zapojit do jeho vývoje či vývojářskou komunitu nějak podporovat. Není to však podmínka pro

jeho užití. Pro společnosti může být výhodné uvolnit software pod open source licencí a tvořit zisk pomocí jeho podpory a školení uživatelů. [5, str. 189]

2.7.2 Typy FOSS licencí

Open source licence jsou často copyleftové. Ty zajišťují přenos svobod a práv původní licence na další uživatele. Každá další verze copyleftového softwaru musí být šířena pod copyleftovou licencí. Tyto licence se dále dělí na silné a slabé copyleftové licence. Další možné dělení je copyleft plný (vztahující se na všechny části softwaru) a částečný (umožňující některé části upraveného softwaru šířit pod jinou licencí). Open source může být i s permisivní copyfree licencí, která se vyznačuje velmi vysokou mírou svobody. [18, str. 291]

V následujícím seznamu je zjednodušený popis některých FOSS licencí. Pro přesné znění je potřeba přejít k oficiálnímu zdroji jednotlivých licencí. [5, str. 189] [18, str. 291–293]

GNU GPL GNU General Public License je „reciproční“ licence, což zjednodušeně znamená, že při použití našeho kódu s kódem pod touto licencí, musí i náš kód vyjít pod licencí kompatibilní s GNU GPL. Tato licence existuje v několika verzích s různými specifiky. Každý soubor s kódem musí obsahovat označení autora, rok vývoje a označení konkrétní licence, pod kterou se nachází. Autor si může účtovat poplatky za zhotovení softwaru, provedení jeho kopie i za provedení servisu. Na tento software se nevtahuje žádná záruka a autoři nejsou zodpovědní za škody při jeho použití, pokud není škoda úmyslná, či z hrubé nedbalosti.

GNU LGPL GNU Lesser General Public License je varianta GNU GPL licence původně učená pro softwarové knihovny. Narozdíl od GNU GPL je méně copyleftová, což umožňuje psát programy pod libovolnou licencí, které na software pod LGPL pouze odkazují. Provedené změny v kódu pod GNU LGPL je však nutné znovu publikovat pod touto licencí.

MPL Mozilla Public Licence staví na povinnosti sdělovat dalším nabyvatelům, že jde o software pod touto licencí a musí být na ní přímý odkaz. Tato licence pochází od společnosti Mozilla.

Modifikovaná BSD Někdy označována jako tříbodová Berkeley Software Distribution je necopyleftová licence. Vyžaduje pouze zachování jména autora, rok vytvoření či pozměnění softwaru a zřeknutí se záruky.

Apache Apache licence je permisivní, pocházející od Apache Software Foundation. Nevyžaduje šíření modifikací produktu pod stejnou licencí, vylučuje bezplatné záruky. Druhá verze této licence je kompatibilní s GPL verze 3.

AGPL Affero General Public Licence nařizuje, aby jakýkoli software využívající komponenty licencované pod AGPL byl i on sám pod AGPL a zároveň aby byl volně dostupný zdrojový kód takové aplikace pomocí internetové sítě. Tato vlastnost ji dělá nejvíce copyleftovou licencí.

Original BSD Berkeley Software Distribution je necopyleftová permisivní licence. Nevyžaduje od vývojářů znovu publikovat změny či úpravy, které provedli v kódu pod touto licencí. Kód pod touto licencí může být součástí softwaru pod jakoukoli licencí. Takový kód lze zahrnout i do proprietárních systémů. Jedinou podmínkou je odkázat na původního autora použitého softwaru a informace o licenci s upozorněném o vyloučení zodpovědnosti.

MIT Massachusetts Institute of Technology licence (též X11 licence) je permisivní licence bez copyleftu (copyfree) s podmínkou, že informace o autorství a vyloučení záruk musí být ve všech kopiích zásadní části softwaru.

Z výše uvedených vlastností jednotlivých licencí vychází, že pro vývoj komerčního produktu lze použít softwarové komponenty pod BSD, MIT, LGPL a MPL licencí. Pokud vyvíjíme produkt pod GPL či AGPL, lze využít komponenty i pod GPL a AGPL licencí.

Sporné je dynamické linkování silně copyleftového softwaru. Existují tři náhledy na tuto problematiku: [18, str. 296–297]

- Aplikace používající dynamicky linkované knihovny je odvozeným dílem a viralita vzniká i při vzájemné komunikaci programů.
- U dynamického linkování není aplikace odvozeným dílem, pokud proprietární software pouze k opensource produktu přistupuje bez úpravy jeho kódu.
- Pokud software není určen ke spojování programů a vytváření odvozených děl, pak odvozené dílo nevzniká bez ohledu na linkování.

Ideálním řešením je vyhnout se kombinování softwaru s copyleftovým softwarem, pokud výsledný produkt nemá mít také copyleftovou licenci.

Kapitola 3

Analýza

Uvnitř kapitoly jsou rozebrány požadavky od správců sítě na výsledný produkt. Jsou zde definovány také typy uživatelů, jímž je produkt určen. Kapitola obsahuje průzkum současných technologií u cílového ISP. Dále kapitola obsahuje analýzu existujících systémů pro řešení daných požadavků. Ta slouží ke zjištění, zda již není potřebný produkt na trhu, což může poskytovatelům ušetřit náklady na vývoj nového produktu. Výsledkem analýzy je analytický model pro návrh systému.

Veškeré diagramy jsou vytvořeny v aplikaci Eclipse Papyrus ™ [19]. Mimo jiné standardy umožňuje tvořit modely v UML 2.5. Pro běh této aplikace je potřeba mít nainstalované JRE (Java Runtime Environment). Z důvodu změny licence u JRE od společnosti Oracle je využito běhového prostředí OpenJDK. OpenJDK je pod licencí GPL 2.0, která umožňuje použití aplikace i v komerčním prostředí [20].

3.1 Analýza požadavků

Cílem je definování požadavků správců sítě. Požadavky jsou rozděleny na funkční a nefunkční. Funkční požadavky vyjadřují chování systému zadáním vstupních a výstupních podmínek. Nefunkční požadavky vykazují jakostní znaky, jako použitelnost, spolehlivost, výkonnost, udržitelnost a rozšiřitelnost. [4]

Při komunikaci se správci sítě byly autorem práce stanoveny hlavní požadavky, které systém musí splňovat.

3.1.1 Funkční požadavky

F1 Přihlašování uživatelů Systém bude zobrazovat přihlášenému správci veškerá data. Přihlášenému klientovi bude zobrazovat pouze jemu patřící data.

F2 Evidence klientů Správce bude schopen vytvářet, číst, upravovat a mazat záznamy o klientech. Bude možné vyhledávat uživatele v seznamu pomocí filtrů. Klient bude moci zobrazit profil se svými informacemi. U každého klienta bude možné evidovat smlouvu o poskytovaných službách v digitální podobě.

F3 Evidence plateb klientů V systému budou uchovávány historie plateb za služby klientům. Správce bude moci vytvářet, číst, upravovat a mazat záznamy ze správcovského profilu. Bude také moci vyhledat platbu podle zadaných filtrů a vyhledat záznam o klientovi z vybrané platby. Systém bude také umožňovat dávkový import plateb ze souboru CSV ve specifikovaném formátu. Před importem systém zobrazí správci náhled importovaných dat. Klient bude moci informace o jeho platbách pouze číst.

F4 Evidence zařízení u klienta Správce bude moci vytvářet, číst, upravovat a mazat záznamy o zařízeních nacházejících se u klienta. Evidence bude obsahovat informaci, zda je zařízení v nájmu či je ve vlastnictví klienta. Pokud se jedná o síťové zařízení, bude uvedena i jeho IP adresa. K zařízením bude možné psát poznámky. Bude moci vyhledat fyzickou adresu, kde se zařízení nachází.

F5 Nastavení parametrů připojení Správce bude moci nastavovat limity na síťovém zařízení u klienta, pokud v zařízení běží operační systém RouterOS.

F6 Uživatelské rozhraní Systém bude možné ovládat přes GUI pomocí myši a klávesnice nebo na dotykovém zařízení. Uživatelské rozhraní bude využívat webového prohlížeče. Bude lokalizované do českého jazyka.

F7 Základní diagnostika sítě Systém bude správci umožňovat diagnostiku stavu sítě pomocí měření odezvy a trasování mezi serverem, na kterém bude systém nasazen, a zvolenou cílovou adresou. Výsledek bude zobrazovat v přehledném grafu.

3.1.2 Nefunkční požadavky

N1 Rozšiřitelný systém Systém bude upravitelný a rozšiřitelný o další funkce.

N2 Ve webovém prohlížeči Klienti a správci budou moci ovládat systém z webového prohlížeče. K používání aplikace bude postačující prohlížeč s podporou HTML5, CSS3 a ES5. Pro přenos dat bude použit protokol HTTP nebo WebSocket a komunikace bude na TCP portu 80 nebo 443.

N3 Operační systém na serveru Aplikaci bude možné provozovat na serveru s operačním systémem MS Windows x64 verze 7 Service Pack

1 a vyšší, MS Windows Server x64 verze 2008 a vyšší nebo Debian x64 verze 7 a vyšší.

N4 Zabezpečený přístup Přístup do aplikace bude zabezpečen pomocí identifikátoru uživatele a hesla. Nepřihlášená osoba nebude moci vytvářet, číst, měnit ani mazat citlivá data. Nepřihlášená osoba bude moci zobrazit pouze úvodní stránku systému.

N5 Zabezpečená komunikace Přenášená data po veřejném komunikačním kanálu budou chráněná proti odposlechu šifrováním alespoň na úrovni AES192.

N6 Uchovávání hesel V databázi nebudou uchovávána hesla v plaintextu, ale pouze hash alespoň vytvořený algoritmem SHA256 nebo vyšším.

N7 Databázový systém Databáze s uloženými daty bude oddělitelná od aplikace. Datový soubor nebude pevně svázán s aplikací a zvolený databázový systém bude možné v případě potřeby změnit bez nutnosti měnit celou aplikační logiku.

N8 Víceuživatelský systém Systém bude schopen bez zvýšení odezvy obsluhovat jednotky uživatelů. Při paralelním připojení desítek uživatelů bude systém schopen zpracovat požadavek do jedné sekundy od jeho přijetí na straně serveru.

3.2 Případy užití systému

Z požadavků na informační systém lze odvodit, že v systému budou existovat dvě role uživatelů. Role „Admin“ označuje správce či administrátory sítě a role „Client“ označuje klienty ISP.

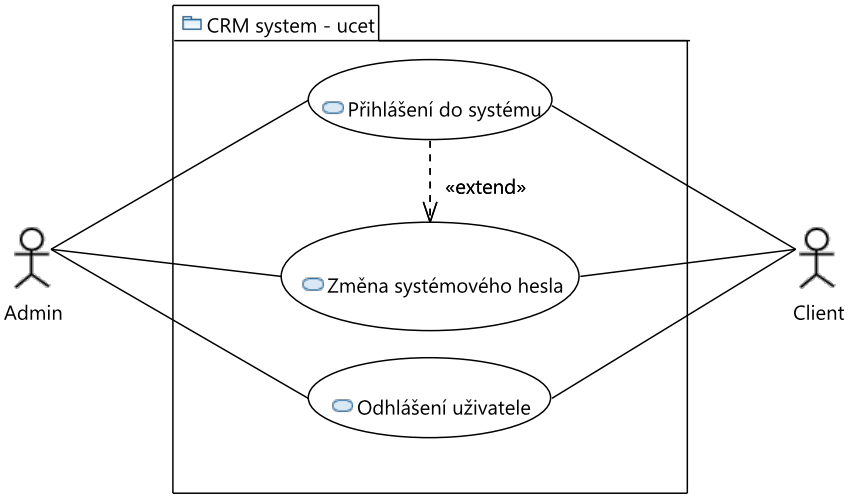
Možnosti každé role jsou zobrazeny na diagramech případů užití (UC). Na základě těchto případů bude systém testován. Testy podle UC poslouží jako důkaz, že jsou požadavky na systém splněny. Pro zlepšení přehlednosti jsou případy rozděleny do několika skupin.

V následujícím seznamu jsou popsány jednotlivé scénáře případů z diagramu 3.1 (CRM system - ucet).

UC-U1 Přihlášení do systému

Hlavní účastník Uživatel
Zúčastněné strany a zájmy

- Libovolný uživatel
- Klientská část systému
- Serverová část systému
- Databáze



Obrázek 3.1: UC nad systémovými účty

Předpoklady

- Funkční připojení k systému a databázi
- Nepřihlášený uživatel nacházející se na přihlašovací obrazovce systému

Scénář úspěšného plnění

1. Uživatel vybere roli pod kterou se chce přihlásit
2. Uživatel vyplní uživatelské jméno či jiný povolený identifikační údaj a heslo
3. Uživatel klikne na tlačítko pro přihlášení
4. Klientská aplikace odešle přihlašovací údaje serverové aplikaci
5. Serverová aplikace nalezne uživatele podle jeho jména a vybrané role v DB a načte jeho sůl a hash hesla
6. Serverová aplikace vytvoří hash pomocí soli z DB a hesla zadaného uživatelem
7. Serverová aplikace porovná vytvořený hash a hash z DB, pokud se shodují, načte z DB ID relace
8. Serverová aplikace zjistí zda je načtené ID relace platné, pokud není, vytvoří nové a uloží ho do DB
9. Serverová aplikace odešle ID relace klientské aplikaci
10. Klientská aplikace uloží ID relace a přesměruje uživatele na úvodní obrazovku

Výjimky

- Neplatné heslo
- Uživatel nenalezen
- Systémová chyba s jejím popisem

UC-U2 Změna systémového hesla

Hlavní účastník Uživatel
Zúčastněné strany a zájmy

- Uživatel požadující změnu hesla
- Klientská část systému
- Serverová část systému
- Databáze

Předpoklady

- Funkční připojení k systému a databázi
- Přihlášený uživatel se nachází na úvodní obrazovce

Scénář úspěšného plnění

1. Uživatel klikne na tlačítko pro změnu hesla
2. Klientská část systému zobrazí okno s formulářem pro zadání původního a nového hesla
3. Uživatel formulář vyplní
4. Klientská aplikace ověří shodu nových hesel
5. Data z formuláře klientská aplikace odešle data na server
6. Serverová aplikace ověří shodu původního hesla s aktuálně uloženým v DB
7. Serverová aplikace vygeneruje náhodnou sůl a společně s ní vytvoří hash hesla
8. Serverová aplikace uloží novou sůl a hash do DB a odešle zprávu klientské aplikaci informaci o úspěšné operaci
9. Klientská aplikace zavře okno pro změnu hesla a oznámí uživateli informaci, že bylo heslo změněno

Výjimky

- Neplatné původní heslo
- Nová hesla se neshodují
- Nové heslo nesplňuje požadavky na délku nebo obsažené znaky
- Systémová chyba s jejím popisem

UC-U3 Odhlášení uživatele

Hlavní účastník Uživatel
Zúčastněné strany a zájmy

- Uživatel se chce odhlásit ze systému
- Klientská část systému
- Serverová část systému
- Databáze

Předpoklady

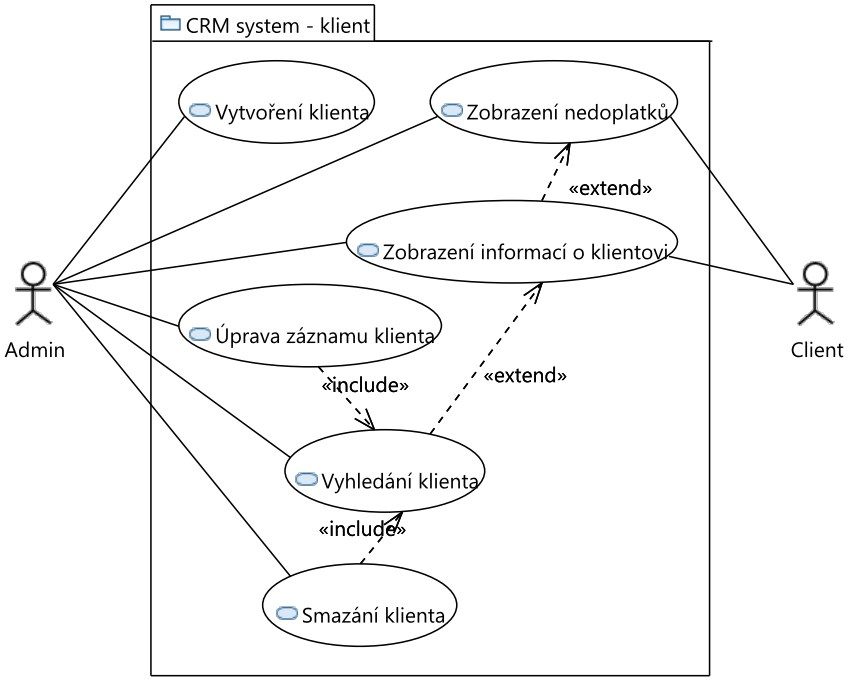
- Funkční připojení k systému a databázi
- Přihlášený uživatel nacházející se na úvodní obrazovce

Scénář úspěšného plnění

1. Uživatel klikne na tlačítko pro odhlášení
2. Klientská aplikace zašle požadavek serverové aplikaci pro odstranění uživatelské relace
3. Serverová aplikace odstraní z databáze uložené ID relace a odešle klientské aplikaci informaci o odstranění
4. Klientská část odstraní uloženou relaci
5. Klientská část odstraní uložené ID relace z prohlížeče a zobrazí uživateli přihlašovací stránku systému

Výjimky

- Systémová chyba s jejím popisem



Obrázek 3.2: UC nad záznamy o klientech

V následujícím seznamu jsou popsány jednotlivé případy z diagramu 3.2 (CRM system - klient).

UC-K1 Vytvoření klienta Správce může vytvořit záznam o novém klientovi. Tento případ bude využit při uzavření smlouvy a zapojení klienta do sítě. V této fázi systém nastaví uživateli aktuální nedoplatek na nulovou hodnotu. Správce zadá adresu klienta na které dochází k čerpání služeb.

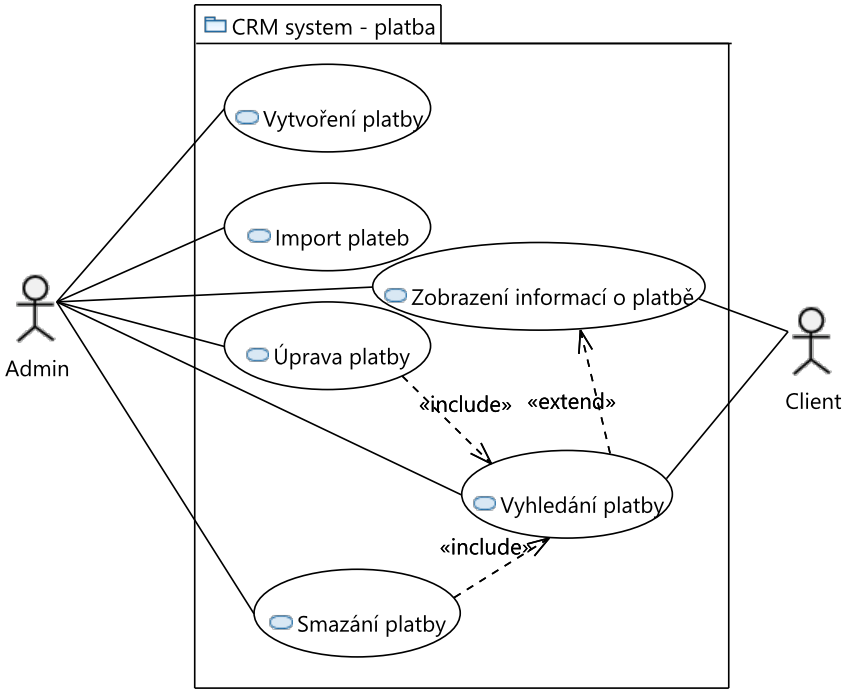
UC-K2 Úprava záznamu klienta Správce má možnost upravit záznam o klientovi. Pomocí vyhledávání v seznamu klientů bude správci usnadněn výběr konkrétního klienta.

UC-K3 Vyhledání klienta Správce bude mít k dispozici kompletní seznam klientů v síti. Aby byla jejich správa snazší, bude možné filtrovat data nad každým atributem klienta.

UC-K4 Smazání klienta Systém umožní správci záznam o klientovi smazat. Z důvodu provázanosti informací o klientech budou předtím odstraněny veškeré záznamy týkající se klienta. Systém na tuto vlastnost upozorní a bude vyžadovat znovu heslo správce, aby nedošlo k neúmyslnému smazání dat o klientech.

UC-K5 Zobrazení informací o klientovi Informace o všech klientech budou dostupné pouze správcům. Klient bude moci zobrazit informace týkající se pouze jeho osoby, ostatní mu nebudou přístupné.

UC-K6 Zobrazení nedoplatků Součástí informací o klientovi bude položka nedoplatky. Ta bude tvořena agregací všech nezaplacených pohledávek. Bude možné zobrazit, které pohledávky jsou nezaplacené.



Obrázek 3.3: UC nad záznamy o platbách

Následuje popis UC z diagramu 3.3 (CRM system - platba).

UC-P1 Vytvoření platby Správce bude mít možnost vytvářet požadavek na platbu i potvrzení o platbě. Požadavky na platbu budou moci být označeny jako periodické. Pokud bude obsahovat koncové datum, opakovat se budou automaticky před uplynutím tohoto data.

UC-P2 Import plateb Platby může správce importovat dávkově z CSV souboru z bankovního výpisu. Pokud bude soubor obsahovat chyby, systém vytvoří upozornění a import neprovede. Pokud budou existovat kolize, dá systém správci na výběr, zda chce ponechat původní platby, nebo je chce přepsat.

UC-P3 Úprava platby Správce bude mít možnost každou platbu ručně upravit.

UC-P4 Smazání platby Správce bude moci platbu ze systému smazat.

UC-P5 Vyhledání platby Pro snadnější správu plateb bude systém poskytovat vyhledávání. Správce bude moci vyhledávat mezi všemi platbami pomocí aplikování různých filtrů. Klient bude moci vyhledávat pouze v platbách, které souvisí s jeho osobou, k ostatním platbám nebude mít přístup.

UC-P6 Zobrazení informací o platbě Správce bude moci zobrazit jakoukoli platbu v systému. Klient bude mít přístup pouze k klatbám související s jeho osobou.

Následuje popis UC z diagramu 3.4 (CRM system - zarizeni).

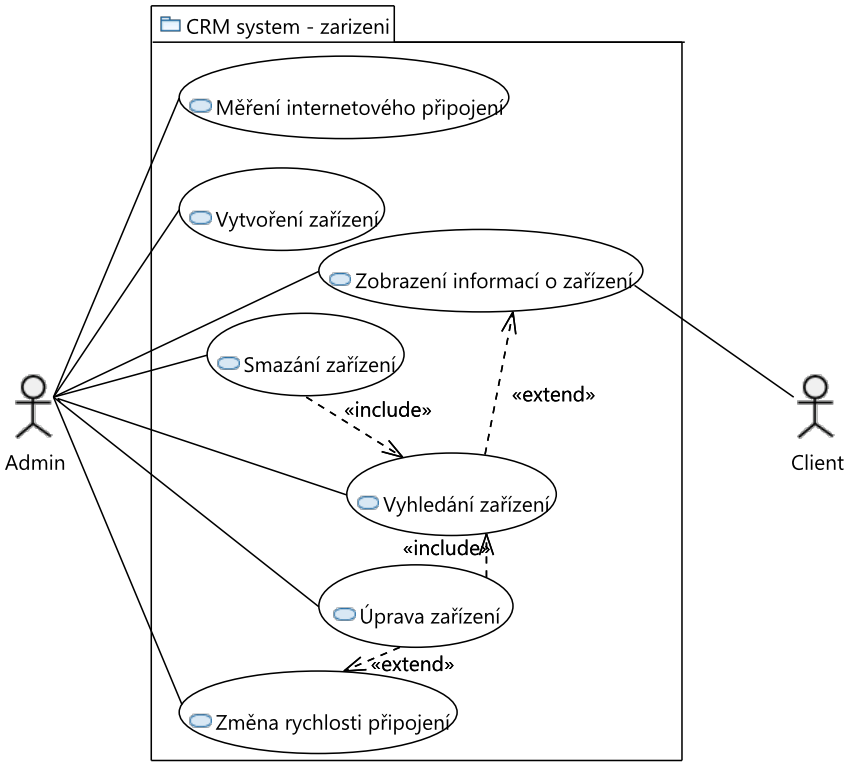
UC-Z1 Vytvoření zařízení Správce bude moci vytvářet záznamy o zařízení. To přiřadí konkrétnímu klientovi. Bude rozlišováno, zda je zařízení klienta, nebo zda je v nájmu a za jaký paušální poplatek, který bude připočítáván k periodickému vyúčtování.

UC-Z2 Úprava zařízení Správce bude moci upravovat uložené záznamy o zařízeních.

UC-Z3 Smazání zařízení Správce bude moci mazat záznamy o zařízeních.

UC-Z4 Zobrazení informací o zařízení Správce bude mít přístup ke všem informacím o zařízeních. Klient bude mít přístup pouze k informacím o jeho zařízeních.

UC-Z5 Změna rychlosti připojení Správce bude moci upravovat rychlost klientských zařízení. Bude moci nastavovat omezení na upload i download.



Obrázek 3.4: UC nad záznamy o klientských zařízeních

UC-Z6 Vyhledání zařízení Správce bude mít možnost pomocí filtrů vyhledávat v seznamu všech zařízení.

UC-Z7 Měření internetového připojení Správce bude moci měřit rychlost a stabilitu připojení klientů, pokud na zařízení běží operační systém RouterOS.

3.2.1 Zobecněné scénáře podobných případů užití

V této části jsou zobecněné scénáře případů užití, které se často pro jednotlivé UC opakují. Tyto UC se liší pouze tím, nad jakými záznamy se operace provádějí.

Scénář pro tvorbu nových záznamů

Hlavní účastník Zaměstnanec
Zúčastněné strany a zájmy

- Zaměstnanec chce přidat nový záznam do systému
- Klientská část systému

- Serverová část systému
- Databáze

Předpoklady

- Funkční připojení k systému a databázi
- Přihlášený zaměstnanec v roli admin na úvodní stránce

Scénář úspěšného plnění

1. Uživatel klikne na tlačítko pro přidání požadovaného záznamu
2. Klientská aplikace zobrazí formulář s požadovanými atributy
3. Uživatel vyplní atributy a stiskne tlačítko pro uložení záznamu
4. Klientská aplikace zkontroluje, zda hodnoty ve vyplněných polích odpovídají požadavkům
5. Klientská aplikace zavolá API serverové aplikace a předá ji hodnoty z formuláře
6. Serverová aplikace zkontroluje, zda jsou data validní
7. Serverová aplikace uloží data do databáze
8. Serverová aplikace odešle klientské aplikace informaci o provedení operace
9. Klientská aplikace zobrazí uživateli hlášku o úspěšném provedení operace a zobrazí okno s detailními informacemi o záznamu

Výjimky

- Záznam již existuje
- Neplatná hodnota v poli
- Systémová chyba s jejím popisem

Scénář pro úpravu existujících záznamů

Hlavní účastník Zaměstnanec
Zúčastněné strany a zájmy

- Zaměstnanec chce upravit existující záznam v systému
- Klientská část systému
- Serverová část systému
- Databáze

Předpoklady

- Funkční připojení k systému a databázi
- Přihlášený zaměstnanec v roli admin na úvodní stránce

Scénář úspěšného plnění

1. Uživatel klikne na tlačítko pro zobrazení seznamu existujících záznamů vybraného typu
2. Klientská aplikace zobrazí okno s formulářem pro filtrování obsahu a seznamem záznamů, které vyhovují zadaným filtrům
3. Uživatel zadá filtry ke zjednodušenému výběru požadovaného záznamu a klikne na tlačítko pro filtrování obsahu

4. Klientská aplikace odešle požadavek na serverovou aplikaci se zadanými filtry
5. Serverová aplikace získá z databáze záznamy, které vyhovují zadaným filtrům
6. Serverová aplikace odešle výsledné hodnoty klientské aplikaci
7. Klientská aplikace zobrazí v seznamu záznamů získaná data
8. Uživatel klikne na požadovaný záznam
9. Klientská aplikace zobrazí okno s detailními informacemi o vybraném záznamu
10. Uživatel klikne na tlačítko pro úpravu požadovaného záznamu
11. Klientská aplikace zobrazí formulář s požadovanými atributy, které budou vyplněné podle aktuálního stavu záznamu
12. Uživatel upraví atributy a stiskne tlačítko pro uložení
13. Klientská aplikace zkontroluje, zda hodnoty ve vyplněných polích odpovídají požadavkům
14. Klientská aplikace zavolá API serverové aplikace a předá ji hodnoty z formuláře
15. Serverová aplikace zkontroluje, zda jsou data validní
16. Serverová aplikace uloží data do databáze
17. Serverová aplikace odešle klientské aplikace informaci o provedení operace
18. Klientská aplikace zobrazí uživateli hlášku o úspěšném provedení operace a zobrazí okno s detailními informacemi o záznamu

Výjimky

- Požadovaný záznam neexistuje
- Neplatná zadaná hodnota
- Systémová chyba s jejím popisem

Scénář pro odstranění záznamů

Hlavní účastník Zaměstnanec

Zúčastněné strany a zájmy

- Zaměstnanec chce odstranit záznam ze systému
- Klientská část systému
- Serverová část systému
- Databáze

Předpoklady

- Funkční připojení k systému a databázi
- Přihlášený zaměstnanec v roli admin na úvodní stránce

Scénář úspěšného plnění

1. Uživatel klikne na tlačítko pro zobrazení seznamu existujících záznamů vybraného typu
2. Klientská aplikace zobrazí okno s formulářem pro filtrování obsahu a seznamem záznamů, které vyhovují zadaným filtrům

3. Uživatel zadá filtry ke zjednodušenému výběru požadovaného záznamu a klikne na tlačítko pro filtrování obsahu
4. Klientská aplikace odešle požadavek na serverovou aplikaci se zadanými filtry
5. Serverová aplikace získá z databáze záznamy, které vyhovují zadaným filtrům
6. Serverová aplikace odešle výsledné hodnoty klientské aplikaci
7. Klientská aplikace zobrazí v seznamu záznamů získaná data
8. Uživatel klikne na požadovaný záznam
9. Klientská aplikace zobrazí okno s detailními informacemi o vybraném záznamu
10. Uživatel klikne na tlačítko pro odstranění požadovaného záznamu
11. Klientská aplikace zavolá API serverové aplikace pro odstranění požadovaného záznamu
12. Serverová aplikace odstraní záznam z databáze
13. Serverová aplikace odešle klientské aplikace informaci o provedení operace
14. Klientská aplikace zobrazí uživateli hlášku o úspěšném provedení operace

Výjimky

- Záznam neexistuje
- Systémová chyba s jejím popisem

Scénář pro měření stability připojení

Hlavní účastník Zaměstnanec
Zúčastněné strany a zájmy

- Zaměstnanec chce měřit stabilitu spoje
- Klientská část systému
- Serverová část systému
- Zařízení s RouterOS, ze kterého bude měření provedeno

Předpoklady

- Funkční připojení k systému
- Funkční připojení k požadovanému síťovému zařízení
- Přihlášený zaměstnanec v roli admin na úvodní stránce

Scénář úspěšného plnění

1. Uživatel klikne na tlačítko pro měření spojů
2. Klientská aplikace zobrazí okno s možnými nástroji pro měření
3. Uživatel klikne na požadovaný nástroj
4. Klientská aplikace zobrazí formulář pro zadání požadovaných parametrů pro měření
5. Uživatel vyplní formulář a klikne na tlačítko pro spuštění měření

6. Klientská aplikace naváže se serverem obousměrný komunikační kanál a odešle přes něj požadavek na měření
7. Serverová aplikace přijme požadavek, připojí se na základě získaných parametrů do síťového zařízení a spustí měření
8. Síťové zařízení průběžně odesílá data z měření serverové aplikaci
9. Serverová aplikace přes otevřený kanál odesílá klientské aplikaci data z měření
10. Klientská aplikace zobrazuje data v uživatelsky čitelné formě
11. Po ukončení měření se serverová aplikace odpojí ze síťového zařízení a odešle klientské aplikaci informaci o odpojení
12. Klientská aplikace uzavře komunikační kanál se serverovou aplikací a zobrazí uživateli informaci o dokončení

Výjimky

- Chyba přihlášení do zařízení
- Chyba spojení
- Neplatný parametr
- Systémová chyba s jejím popisem

3.3 Průzkum technologií a principů u ISP

Aby bylo možné vytvořit návrh řešení systému, bylo potřeba provést analýzu technologií, které cílový ISP používá. Jelikož je autor práce již několik let asistentem malého ISP, má přehled o tom, kam takové společnosti směřují svůj byznys a jak jej rozvíjejí pomocí moderních technologií.

Nejčastěji má síť ISP několik přípojných bodů do páteřních sítí v republice. Celá vnitřní síť ISP je nastavená jako lokální. V síti poskytovatele jsou různé typy zařízení. Hlavní členění je podle možnosti konfigurovatelnosti. Konfigurovatelná zařízení mají vlastní IP adresu. K nim je možné se připojit a nastavovat je. Pokud taková zařízení disponují pokročilejším vlastním systémem, je možné na nich provozovat měření síťových spojů. ISP mají často zařízení od různých výrobců. Každý výrobce má vlastní nástroje pro nastavení svých zařízení.

ISP u koncových klientů vytvářejí lokální sítě, které jsou od sítě ISP logicky oddělené. Někteří koncoví klienti mají vlastní zařízení pro připojení do sítě ISP a někteří mají zařízení sdílená, ve kterých se dělí síť ISP a klienta. Za těmito zařízeními se již nachází lokální síť koncového klienta. V takové síti se mohou nacházet libovolná síťová zařízení od různých výrobců, do kterých ISP nezasahuje. Taková zařízení ISP neeviduje, jsou ve vlastnictví klienta a pokud si to klient výslovně nevyžádá, nechá nastavení na klientech.

ISP buduje infrastrukturu k zajištění datového spojení mezi páteřními spoji a zařízeními na rozhraní ISP a koncových klientů. Infrastruktura je budována pomocí drátových a bezdrátových spojů. U bezdrátových spojů je problém s okolním rušením od konkurenčních ISP. Proto se v místech kde je to možné

upřednostňují drátová spojení. Většina uzlů v síti obsahuje konfigurovatelné síťové zařízení. Na takových uzlech lze sledovat datové propustnosti, kvalitu spojení, omezovat rychlost datových toků atd. Pro zajištění kvality spojení je potřeba zajišťovat nepřetržitý běh a kontrolovat stav takových zařízení.

V infrastruktuře ISP jsou nejčastějšími výrobci síťových zařízení MikroTik a Ubiquiti. Společnost MikroTik má operační systém RouterOS, který je určený pro jejich zařízení. Je možné RouterOS zakoupit i jako samostatnou licenci a provozovat ji na zařízení, které RouterOS podporuje. RouterOS je možné konfigurovat pomocí příkazové řádky, přes webové rozhraní a pomocí aplikace WinBox.

Síťové zařízení může disponovat různými počty rozhraní, na které se vážou IP adresy. Pomocí IP adres zařízení spolu komunikují. V lokální síti ISP se běžně používá IPv4. Některá zařízení jsou nastaveny jako obyčejné přepínače, některá obsahují i nakonfigurované směrovací tabulky. Některá zařízení jsou modulární, některá neumožňují žádné rozšíření. U koncových klientů ISP nejčastěji volí AIO řešení pro nižší cenu a snadné uvedení do provozu. Hlavní uzly obsahují komplikovanější zařízení, aby zvládalo vyšší rychlosti datových toků. Protože sítě jsou průběžně zlepšovány, obsahuje různě stará zařízení disponující různými protokoly a různými verzemi operačních systémů.

Jelikož se u ISP nasazují aplikace pro sběr a analýzu dat ze sítě, pro podporu administrace síťových prvků a mnoho dalších aplikací, nasazují se na servery v hlavních uzlech. Na takových serverech je běžně OS Linux. Ty se zároveň používají pro automatické zálohy konfigurací se síťových prvků. Jelikož je toto řešení centralizované, lze levným způsobem takové zařízení duplikovat a tím jej mít zálohované.

3.4 Průzkum existujících řešení

CRM systémů pro ISP existuje mnoho. Většina z nich je komerční produkt bez možnosti úprav na míru cílovému ISP. V řadě případů tyto systémy slouží pouze pro evidenci klientů a řízení platebních procesů. Pro malé ISP jsou existující řešení často nevýhodná z důvodu nemožného rozšíření podle vlastních požadavků či z důvodu vysoké pořizovací ceny. Níže jsou popsány pouze systémy, které dokáží alespoň nějakým základním způsobem pracovat se zařízeními od společnosti MikroTik, jsou dostupné v českém nebo anglickém jazyce a splňují alespoň hlavní požadavky na systém.

ExpertBilling [21] je fakturační systém umožňující spolupráci se zařízeními s RouterOS. Tato webová aplikace je napsaná v jazyce Python. Poskytuje management faktur, sběr dat a tvorbu statistik z provozu sítě. Tento produkt je komerční a lokalizovaný pouze do ruského jazyka.

Splynx ISP Framework [22] je systém pro WISP s fakturačními funkcemi vybaven s centrálním konfiguračním a monitorujícím portálem sítě. Jedná

se o velmi propracované řešení. Díky modularitě systému jej lze rozšířit o komunikační rozhraní pro jiné systémy a zároveň disponuje vlastním API. Obsahuje rozhraní pro zařízení MikroTik i Ubiquiti, díky kterému umožňuje management datových spojů. Cena za jeho provoz je v současné době 0,5 $ na klienta za měsíc. Pro malé ISP jsou tedy náklady v řádu stovek dolarů měsíčně.

MUPSSOFT [23] je produkt pro správu zařízení s RouterOS. Mezi jeho funkce patří zálohování zařízení, sledování datových toků, konfigurace firewallu, tvorba logů stavu sítě atd. Není však určen ke komunikaci s klienty a jejich management.

IconWave [24] poskytuje řešení pro velké ISP. Produkty této společnosti poskytují automatické vyúčtování, řízení datové propustnosti, grafickou vizualizaci aktuálního stavu sítě, automatickou evidenci událostí v síti, evidenci klientů, monitoring datových toků, automatické upozornění na výpadek zařízení a mnoho dalších funkcionalit.

ISP Cube [25] umožňuje elektronickou tvorbu faktur za služby, kontrolovat datový tok v síti, instalaci na zařízení v cloudu i u ISP, klientům umožňuje platit za služby pomocí platebních portálů, automatické odpojení klientů od služeb při nezaplacení. Náklady na provoz produktu jsou pro malé ISP v řádu stovek $ měsíčně. Společnost umožňuje úpravy produktu podle požadavků ISP.

Magnaquest SURE! [26] má v nabídce produkt „SURE! Broadband Billing & CRM System“. Podporuje drátové i bezdrátové technologie u ISP. Poskytuje řešení pro řízení plateb, kontrolu kvality dodávaných služeb. Tento produkt je určený spíše větším ISP a mobilním operátorům.

Anatod [27] nabízí produkt pro řízení plateb klientů, komunikaci se síťovými zařízeními od společnosti MikroTik a Ubiquiti, sledování připojených zařízení a datových toků mezi nimi, pomocí Google Maps API je možné zařízení zadávat kde se nacházejí a tvořit tak mapu sítě, detekci a evidenci problémů v síti. Náklady na produkt jsou pro malé ISP od 600 $ měsíčně.

ISPERP [28] je služba v cloudu pro ISP poskytující grafickou vizualizaci reportů ze sítě dostupnou s osobních počítačů i mobilních zařízení, management připojení uživatelů, webový portál pro klienty, fakturační systém s možností SMS notifikace a platební bránou, portál pro hlášení závad na síti ze strany klientů, tvorbu logů ze sítě, a mnoho dalších prvků. Náklady na produkt společnost neuvádí a poskytuje individuální nabídky přesně pro konkrétní ISP.

ISP Billing [29] umožňuje pomocí grafické vizualizace zobrazit přehledně aktuální stav sítě, konfigurovat rychlosti připojení jednotlivých klientů,

nastavovat časové limity a pomocí jeho API je možné propojit jiné systémy s tímto produktem.

Jaze ISP Manager [30] slouží ISP k evidenci klientů, řízení přidělování IP adres, evidenci síťového vybavení, sledování plateb, evidenci a řešení problémů s dostupností služeb, řízení rychlostí připojení (možno i podle cílové domény či použitého protokolu), zasílání notifikací o platbách, sledování stability sítě, přiřazování síťových zařízení ke klientům, sledování historie přesunu zařízení, atd. Pro administrátory poskytuje mobilní aplikaci se základními funkcemi. Umožňuje také tvorbu vizualizací stavu sítě.

V seznamu jsou vybrány a popsány nejzajímavější produkty. Žádný nalezený produkt není lokalizovaný do českého jazyka, což není problém pro administrátorské rozhraní, ale pro rozhraní určené klientům to problém je.

3.5 Analytický model domény

Analytický doménový model na obrázku 3.5 popisuje analyzovanou doménu u ISP. Jednotlivé entity reprezentují skutečné objekty, atributy jejich vlastnosti a vazby vztah mezi nimi. Detailnější informace k diagramu včetně popisu obsahu v atributech jsou rozepsány dále ke každé entitě zvlášť.

3.5.1 Umístění

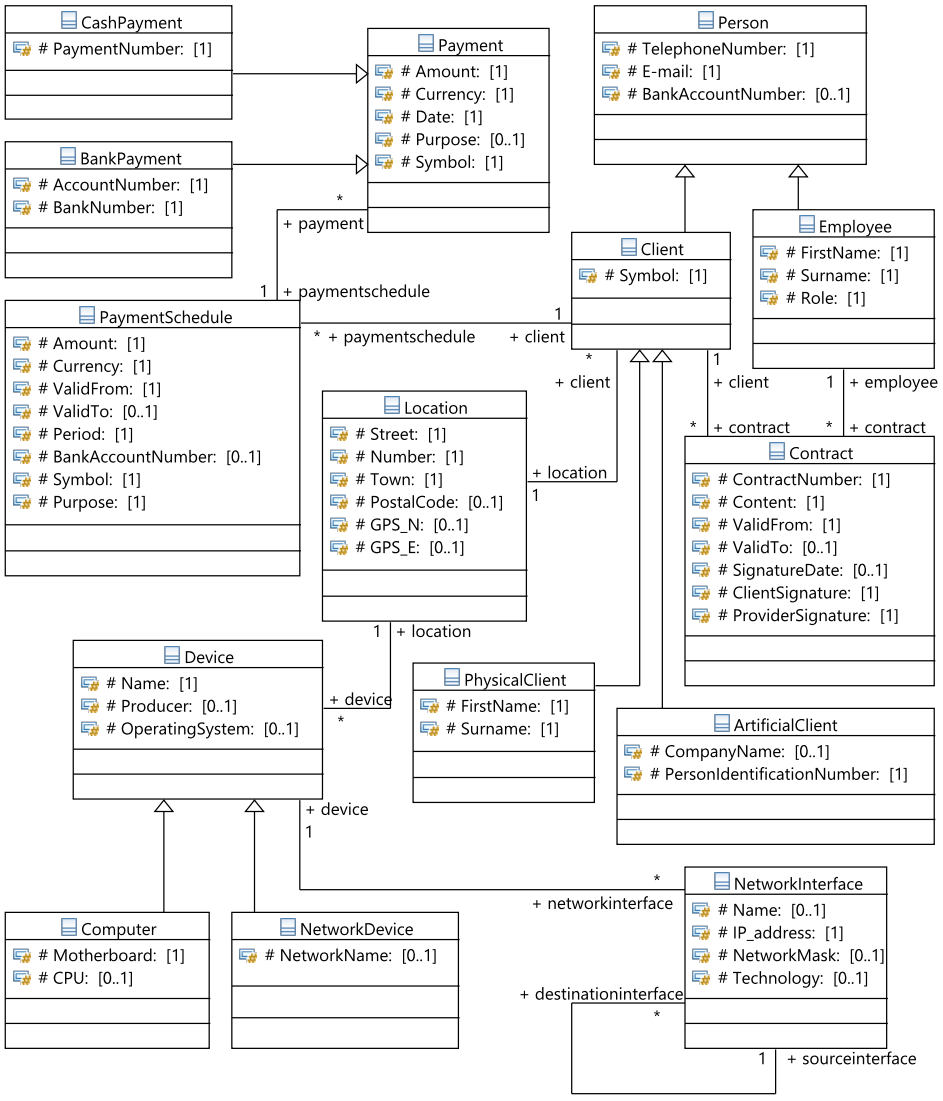
Entita **Location** představuje skutečné umístění objektu.

Street Ulice
Number Číslo popisné / číslo orientační
Town Název města či vesnice
PostalCode Poštovní směrovací číslo
GPS_N Geografická souřadnice poledníku východní polokoule
GPS_E Geografická souřadnice rovnoběžky severní polokoule

3.5.2 Zařízení

Entita **Device** představuje fyzické zařízení.

Producer Výrobce zařízení
Name Oficiální označení zařízení od výrobce
OperatingSystem Operační systém běžící na zařízení



Obrázek 3.5: Analytický doménový model

3.5.2.1 Computer

Zařízení typu **Computer** je jakékoli výpočetní zařízení, které se neúčastní řízení provozu v síti.

Motherboard Základní deska nebo hlavní komponenta v zařízení
CPU Výpočetní čip zařízení

3.5.2.2 Síťové zařízení

Zařízení typu **NetworkDevice** je zařízení určené pro řízení provozu v síti.

NetworkName Virtuální název zařízení v síti

3.5.3 Síťové rozhraní

Entita **NetworkInterface** představuje rozhraní, umožňující propojení jednotlivých zařízení připojitelných do sítě.

Name Uživatelský název rozhraní
IP_address IP adresa nastavená na rozhraní
NetworkMask Maska sítě
Technology Technologie spoje

3.5.4 Osoba

Entita **Person** jakoukoli osobu u ISP. Osoba může být klientem nebo zaměstnancem. Každá osoba má následující atributy.

TelephoneNumber Telefonní kontakt osoby
E-mail E-mailový kontakt osoby
BankAccountNumber Bankovní účet osoby

3.5.4.1 Klient

Entita **Client** dědí vlastnosti z entity Person a rozšiřuje je o další atributy. Reprezentuje klienta poskytovatele.

Symbol Identifikační symbol klienta, kterým se prokazuje při platbách, je vytvořeno při navázání obchodního vztahu a je používáno při platbách a komunikaci s klientem

3.5.4.2 Fyzická osoba

Entita **PhysicalClient** dědí vlastnosti z entity Client a reprezentuje fyzickou osobu.

FirstName Křestní jméno osoby
Surname Příjmení osoby

3.5.4.3 Právnická osoba

Entita **ArtificialClient** dědí vlastnosti z entity Client a reprezentuje právnickou osobu.

CompanyName Název právnické osoby
PersonIdentificationNumber Identifikační číslo

3.5.4.4 Zaměstnanec

Entita **Employee** dědí vlastnosti z entity Person a rozšiřuje je o další atributy. Reprezentuje zaměstnance u ISP.

FirstName Křestní jméno
Surname Příjmení
Role Role u ISP

3.5.5 Smlouva

Entita **Constract** představuje smlouvu o poskytování služeb.

ContractNumber Číslo obchodní smlouvy
Content Obsah smlouvy s informacemi a podmínkách o poskytovaných službách
ValidFrom Datum, ke kterému smlouva nabývá platnosti
ValidTo Datum, ke kterému je sjednaný konec platnosti smlouvy, pokud je smlouva sjednaná na dobu neurčitou, položku nemá vyplněnou
SignatureDate Datum podpisu smlouvy
ClientSignature Podpis klienta
ProviderSignature Podpis poskytovatele služeb

3.5.6 Platební plán

Entita **PaymentSchedule** představuje platební plán za služby.

Amount Sjednaná částka za služby
Currency Měna částky
ValidFrom Datum, ke kterému je vztaženo první vyúčtování
ValidTo Datum, ke kterému bude vztaženo poslední vyúčtování
Period Perioda plateb (jednorázová, týdenní, měsíční, kvartální, roční)
BankAccountNumber Číslo účtu, na který mají platby přicházet
Symbol Identifikační symbol určující plán
Purpose Popis služeb, ke kterým se plán vztahuje

3.5.7 Platba

Entita **Payment** představuje požadovanou nebo přijatou platbu za služby.

Amount Velikost částky
Currency Měna částky
Date Datum vystavení, nebo přijetí platby
Purpose Účel vystavené platby, nebo popis příchozí platby
Symbol Symbol platby k přiřazení ke konkrétnímu plánu

3.5.7.1 Hotovostní platba

Entita **CashPayment** dědí vlastnosti z entity Payment a reprezentuje přijatou hotovostní platbu.

PaymentNumber Číslo stvrzenky předané pro potvrzení přijetí platby

3.5.7.2 Bezhotovostní platba

Entita **BankPayment** dědí vlastnosti z entity Payment a reprezentuje přijatou bezhotovostní platbu.

AccountNumber Číslo účtu, z něhož byla platba přijata
BankNumber Číslo banky, ze které byla platba přijata

3.5.8 Vazby

V této části je popis vazeb mezi entitami.

Device–Location Umístění síťového zařízení
NetworkInterface–Device Síťové rozhraní zařízení
NetworkInterface–NetworkInterface Komunikační spojení mezi síťovými rozhraními
Client–Location Místo, odkud klient odebírá poskytované služby
Contract–Client Smlouva s klientem
Contract–Emplyee Zodpovědný zaměstnanec za smlouvu
PaymentSchedule–Client Platební plán klienta
Payment–PaymentSchedule Platba k platebnímu plánu

Kapitola 4

Návrh řešení

V návrhu jsou popsány jednotlivé technologie, na kterých bude systém postaven. Je v něm vytvořena softwarová architektura, navrženo API a uživatelské rozhraní, vybrány komponenty pro sestavení modulárního webového systému a popsána databáze.

4.1 Výběr vhodných softwarovýh nástrojů pro vývoj IS

Při výběru technologií byl kladen důraz na to, aby byly perspektivní s ohledem na možnou rozšiřitelnost systému. Zároveň byly technologie voleny tak, aby v případě nutnosti výměny některé části nebylo nutné systém zcela přestavět například kvůli omezení nějakým frameworkem.

4.1.1 JavaScript

Při volbě jazyka pro vývoj aplikace bylo vybíráno podle žebříčku popularity a zároveň, aby měl jazyk dostatečně dlouhou historii. Z žebříčků v roce 2019 jsou nejoblíbenější jazyky Python (29,5 %), Java (19,6 %), JavaScript (8,4 %), C# (7,4 %), PHP (6,3 %) a C/C++ (5,9 %). Seznam nejoblíbenějších jazyků je dostupný na adrese `http://pypl.github.io/PYPL.html`.

Nejoblíbenějším je jazyk Python, který je vhodný pro tvorbu jednoduchých aplikací a nevyskytuje se v implementacích větších systémů. Jeho nasazení je vhodné pro datamining. Java je oblíbený jazyk v korporátní sféře, náklady na programátory aplikací v jazyce Java jsou vyšší, než u ostatních jazyků. Třetím nejoblíbenějším jazykem je JavaScript (JS). Volba padla na něj, jelikož je velmi rozšířen, je možné v něm implementovat klientskou i serverovou část a v poslední době jeho obliba roste. Jazyk C# je vhodný pro vývoj aplikací na platformě MS Windows. Obliba jazyka PHP byla dříve veliká, v poslední době jej ze serverové části vytlačuje zmíněný JS. Jazyk C/C++ je vhodný pro

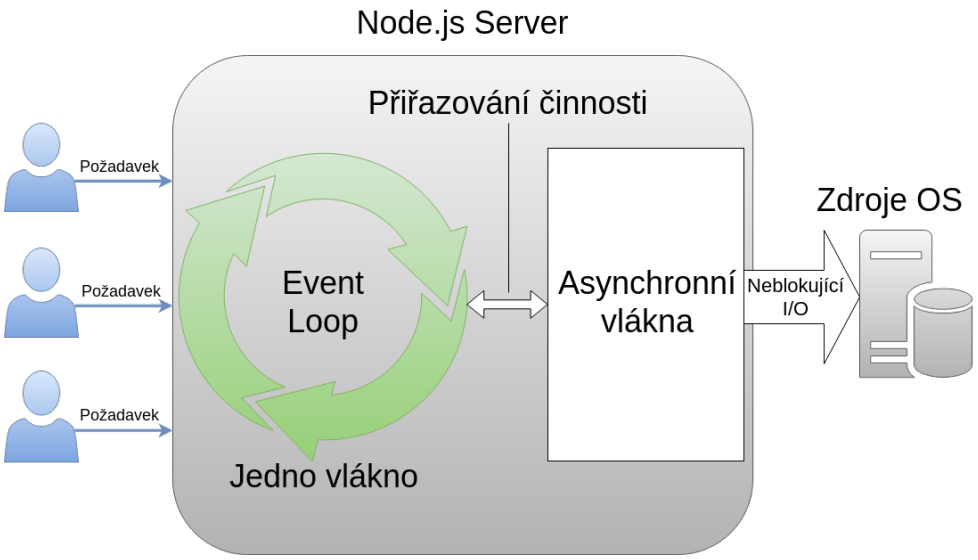
úlohy složité na výpočet, jelikož šetří výpočetní prostředky, ale pro webové aplikace se nepoužívá.

4.1.2 TypeScript

Jako implementační jazyk byl zvolen TypeScript. Jazyk TypeScript (TS) je open–source od společnosti Microsoft, postavený nad jazykem JavaScript. Přináší do JS statické typování, třídy, rozhraní a další vlastnosti. Díky nim předchází chybám programátorů a je považován za bezpečnější než JS. Kód napsaný v TS se následně překládá do JS pomocí transpileru. Zároveň platí, že každý kód napsaný v JS je validním kódem v TS. [31]

4.1.3 Node.js

Pro serverovou část byl zvolen Node.js, který umožňuje spouštět JS kód mimo webový prohlížeč a obsluhovat mnoho požadavků od klientů. Díky architektuře Node.js s použitím smyčky událostí (obrázek 4.1), rozděluje požadavky asynchronním vláknům a neblokuje paralelní zpracovávání více požadavků, pokud jsou pro vyřizování požadavků dostupné potřebné výpočetní a systémové prostředky. [32]



Obrázek 4.1: Smyčka událostí v Node.js [32]

Součástí Node.js je balíčkovací manager **npm**. Díky němu lze v projetu snáze udržovat závislosti na knihovnách a nástrojích. Zároveň usnadňuje aktualizace či přidání prvků do projektu bez složitého zásahu, což podporuje rozšiřitelnost systému.

4.1.4 MongoDB

Při výběru bylo voleno mezi relačními a nerelačními databázemi. Jako vhodnější pro tento systém je dokumentově–orientovaná MongoDB, která patří mezi NoSQL databáze. Ukládá data ve formátu BSON, což je binární JSON. Databázový systém je vydáván pod licencí SSPL (Server Side Public Licence), ta se od GNU AGPL významně liší pouze ve článku 13, který stanovuje, jak zacházet se softwarem pod touto licencí, pokud jej upravíme. Veškeré rozdíly od AGPL popisuje dokument dostupný z `https://webassets.mongodb.com/_com_assets/legal/SSPL-compared-to-AGPL.pdf`. [33]

Pro připojení k databázi a řízení toku dat mezi aplikací a DBS bude použito **mongoose**. Jde o balíček pod licencí MIT `https://www.npmjs.com/package/mongoose`.

4.1.5 Express

Pro tvorbu API na straně serveru využiji komponentu Express. S její pomocí lze snadným způsobem vytvořit API pro obsluhu HTTP požadavků. Existuje možnost rozšíření o protokol websocket, který využívá otevřeného spojení umožňující obousměrnou komunikaci mezi serverem a klientem. Díky němu není nutné používat AJAX a použije se knihovna **Socket.IO**. [34] [35]

4.1.6 Passport.js

Pro autentizaci a autorizaci bude použit Passport.js. Jedná se o middleware, kterým lze snadno rozšířit webovou aplikaci založenou na Express bez složitého zásahu do kódu. Podporuje řadu autentizačních technik, mezi které patří uživatelské jméno a heslo, Facebook, Twitter, OAuth, OpenID, GitHub a mnoho dalších. Po autentizaci vytváří unikátní relaci neboli **session**, ta se ukládá do cookie a uživatel nemusí zadávat přihlašovací údaje při každém požadavku na server. [36] [37]

4.1.7 SASS

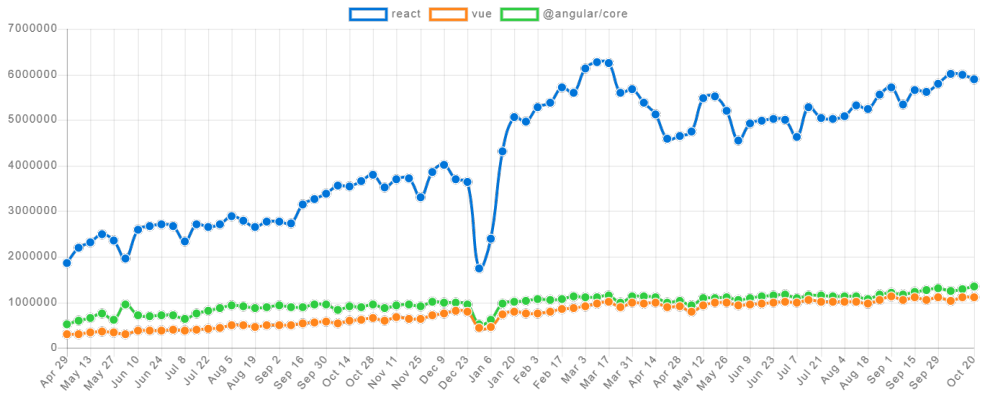
Vzhled webových aplikaci se nastavuje pomocí CSS. Jeho syntaxe však má jistá omezení. Proto v projektu bude použitý SASS, což je CSS preprocesor. SASS je kompilovaný jazyk, který rozšiřuje CSS o proměnné, cykly, podmínky, funkce, aritmetické operace atd. Umožňuje zápis členit do znovupoužitelných souborů, zanořovat kód do sebe, čímž usnadní čitelnost, množství a změnu kódu. Syntaxe kódu je SCSS (Sassy CSS). Platí, že jakýkoli CSS kód je kompatibilní s SCSS. Kód v SCSS se kompiluje do CSS. [38]

4.1.8 Plotly.js

Pro reporting aktuálního stavu sítě a měření vlastností spojů bude využita open–source knihovna Plotly.js. Obsahuje mnoho typů grafů včetně 3D. Je možné s ní tvořit i animované grafy. Je možné ji použít v prohlížečích podporující jazyk SVG. [39]

4.1.9 React

Pro vývoj klientské části u webových aplikací se nejčastěji používá framework Angular, framework Vue, nebo knihovna React. Usnadňují tvorbu interaktivního uživatelského rozhraní v JS. Dnes je velmi rozšířený trend stavby **Single page application** (SPA), které tyto nástroje mění DOM za běhu a aplikace se chováním přibližuje běžné aplikaci.



Obrázek 4.2: Knihovny a frameworky pro UI 2018/04 – 2019/10 [40]

Vybíráno bylo tedy mezi Angular, Vue a React. V poslední době roste obliba React jak ukazuje graf 4.2 v počtech stažení, proto na něj volba padla i v tomto projektu.

React je knihovna v JS pro tvorbu webových komponent. Oproti jQuery se nepíše kód měnící samotný DOM, ale popisuje se, jak má výsledek vypadat a jak má reagovat. Programátor je od DOM odstíněn a v kódu se definuje struktura pomocí skládání React funkcí, pro jejichž zápis se používá JSX syntaxe. React sám na základě definice seskládá virtuální DOM a efektivně zaktualizuje DOM zobrazený uživateli. Z toho důvodu je dnes jQuery na ústupu. React umí i systém pro zpracování událostí a mnoho dalších vlastností. [41]

4.1.10 Babel

Na serverové části lze používat i nové verze ECMAscript (ES), protože víme, jakou verzi Node.js nasadíme. Na straně klienta ale nemáme jistotu, zda používá webový prohlížeč podporující nové verze ES.

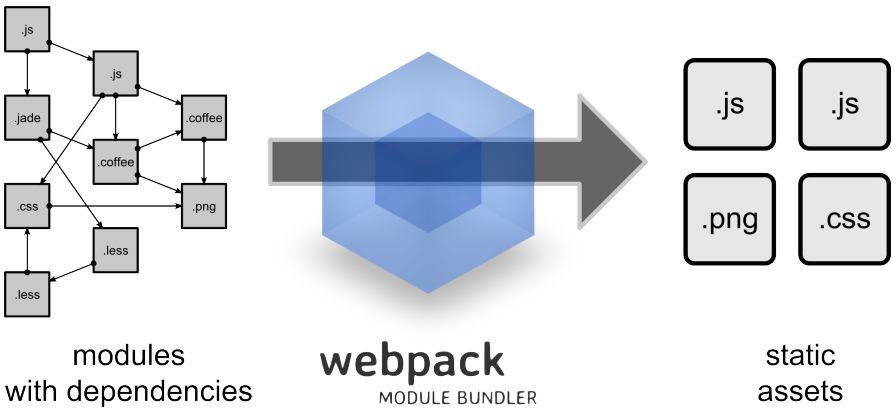
Vývojáři webových aplikací musí zajistit, aby aplikace fungovaly ve všech běžně používaných prohlížečích. Komplikace nastává, pokud chtějí využívat nové vlastnosti ve vyšších verzích ES, které často usnadňují vývoj. [42]

Proto padla volba na použití pro klientskou část kompilátoru Babel a pomocí něj kód pro stranu klienta překládat do ES verze 5. Díky němu lze používat nové vlastnosti v ES. Babel má širokou podporu nových verzí a lze jej použít i v kombinaci s React a jeho JSX syntaxí. [42]

4.1.11 Webpack

V projektu je mnoho prvků, na kterých jsou aplikace závislé. Často se musí při každé změně kódu spouštět nějaké skripty. V projektu jsou závislosti některých nástrojů a jejich pořadí spuštění je nutné dodržet. U velkých projektů je vhodné použít nástroj, který opakované operace, kontrolu závislostí a kontrolu kódu řeší automaticky za programátory. V tomto projektu je potřeba zpracovat kódy z TS, JSX, SCSS a JS.

Těmto požadavkům nejlépe vyhovuje nástroj Webpack. Jeho úkolem je usnadnit práci při vývoji. Jeho základem jsou loadery, které jsou přiřazeny jednotlivým typům souborů. Dále řeší závislosti, na jejichž základě přetransformuje projekt se vším potřebným do koncových balíčků, jak je znázorněno na obrázku 4.3. Na počátku projektu se musí Webpack nakonfigurovat. [43]



Obrázek 4.3: Webpack module bundler [44]

Webpack je vhodný i pro řešení problému s distribucí kódu do cílového prohlížeče přes HTTP. Je potřeba omezit velikost přenášených dat a počet HTTP požadavků. Webpack po programátorech požaduje, aby pečlivě u každé stránky/komponenty/balíčku byly uváděny jejich závislosti na JS/TS modulech, CSS, mediálních souborech, fontech a všech dalších částech pomocí import/require. Webpack potřebuje ještě cestu k počátečnímu prvku, definici

co ignorovat, čím prvky zpracovat a co chceme na výstupu. Tak sestaví graf závislostí a posílat se bude pouze to, co je zrovna potřeba. Jediná nevýhoda Webpack je jeho složitost při počátečním nastavení. [45]

Webpack disponuje funkcí **Hot module replacement**. Ta umí zpracovat moduly a zakomponovat je do výsledné aplikace bez nutnosti obnovení stránky, což se hodí při vývoji, kdy měníme zdrojový kód. To usnadňuje vývoj, jelikož se při změně kódu neztrácí stav aplikace (např. vyplněné formuláře nebo posloupnost kroků v aplikaci). Ne vždy jde tuto funkci použít a při některých změnách je nutné provést obnovení stránky. Při použití React je tato funkce nepoužitelná, pokud je stav uchováván přímo v React komponentách. Řešením je použit **Redux**, který stav komponent uchovává mimo ně a změnou kódu React se stav neztratí. [45]

4.2 Softwarová architektura

V této části je navržena softwarová architektura informačního systému. Popisuje nasazení jednotlivých částí CRM systému a hlavní systémové komponenty s jejich závislostmi.

4.2.1 Diagram nasazení

V diagramu jsou popsány podpůrné systémy a aplikace pro nasazení výsledného systému do produkčního prostředí. Na obrázku 4.4 je zobrazen diagram nasazení systému u ISP.

4.2.1.1 DB server

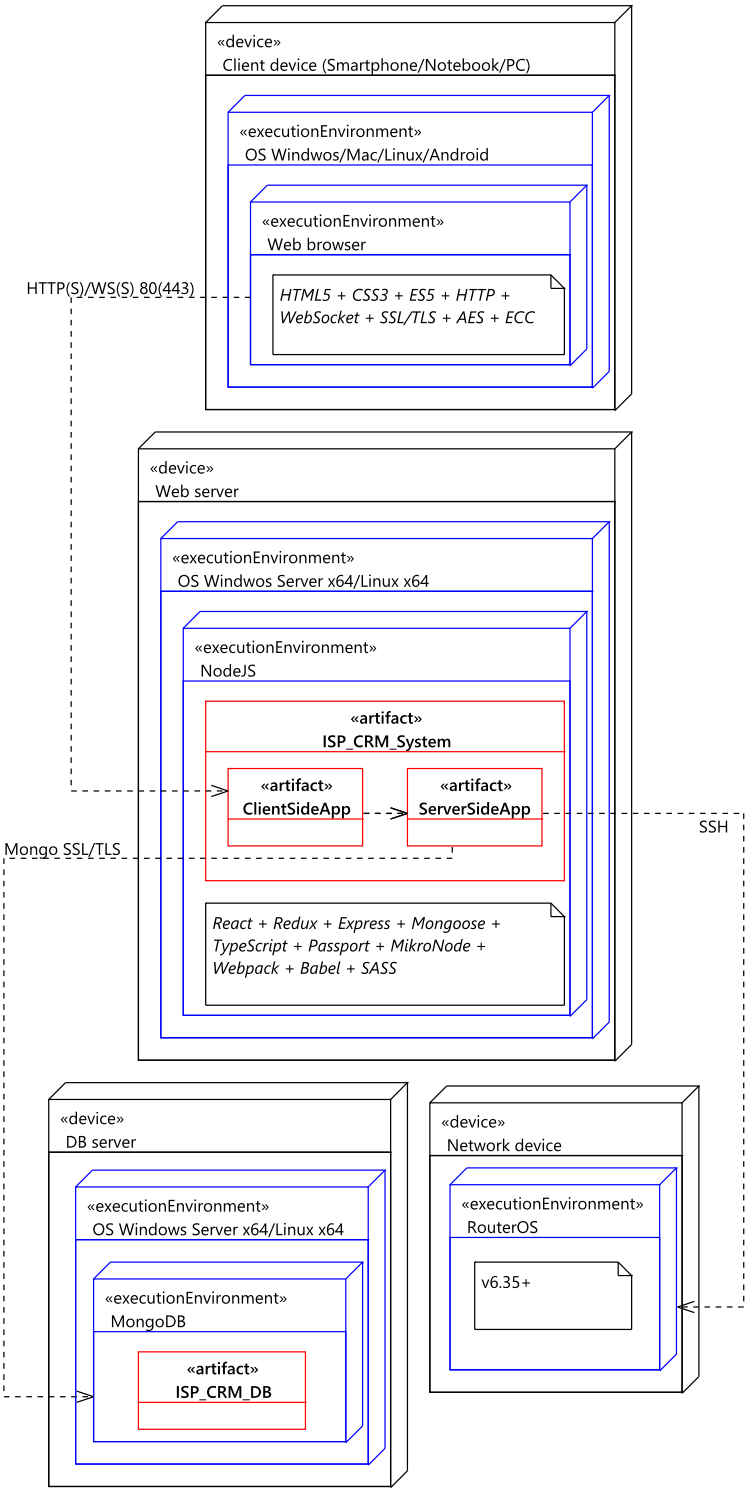
Databáze systému označená jako **ISP_CRM_DB** je určena k ukládání dat z celého CRM systému. Provozována bude na databázovém systému **MongoDB Community Server**. Pro její běh je podmínkou 64 bitový operační systém. MongoDB podporuje OS Windows i mnoho distribucí OS Linux. V ideálním případě bude DB server na nezávislém fyzickém zařízení na aplikačním serveru.

4.2.1.2 NetworkDevice

V síti ISP je mnoho síťových zařízení, na kterých běží operační systém RouterOS. Podmínkou je verze RouterOS 6.35 nebo vyšší pro plnohodnotné využití API.

4.2.1.3 Web server

Webový server je centrální zařízení CRM systému. Bude obsahovat 64 bitový OS Windows nebo Linux. Na něm bude nasazen systém **Node.js**, jehož součástí je správce balíčků **npm**, pomocí kterého bude systém schopen sobě doinstalovat



Obrázek 4.4: Diagram nasazení CRM systému

potřebné balíčky. Jimi jsou React, Redux, Express, Mongoose, TypeScript, Passport, MikroNode, Webpack, Babel a SASS.

Samotný systém označený jako **ISP_CRM_System** se bude skládat ze dvou aplikací. Aplikace **ClientSideApp** poslouží k interakci s uživatelem, transformaci dat do vhodné vizuální podoby a bude optimalizována pro běh na široké škále webových prohlížečů. Aplikace **ServerSideApp** bude obsluhovat požadavky přes její API, získávat data z různých zdrojů, udržovat data přístupná pouze autorizovaným uživatelům, autentizovat uživatele a transformovat data podle interních potřeb.

4.2.1.4 Client device

Zařízení **Client device** představuje uživatelské zařízení typu Smartphone, Notebook nebo PC. Na tomto zařízení je provozován OS Windows, Mac, Linux, nebo Android. V tomto operačním systému je webový prohlížeč. Na něm je možné přistupovat k CRM systému přes webové rozhraní. Webový prohlížeč musí podporovat HTML5, CSS3, ES5, HTTP, WebSocket, SSL/TLS, AES a ECC.

4.2.1.5 Komunikační kanály

Mezi jednotlivými částmi probíhá komunikace pomocí různých kanálů.

ServerSideApp–ISP_CRM_DB Komunikační kanál mezi serverovou částí aplikace a databází bude šifrovaný. Veškeré čtení a zápis dat půjde tímto kanálem.

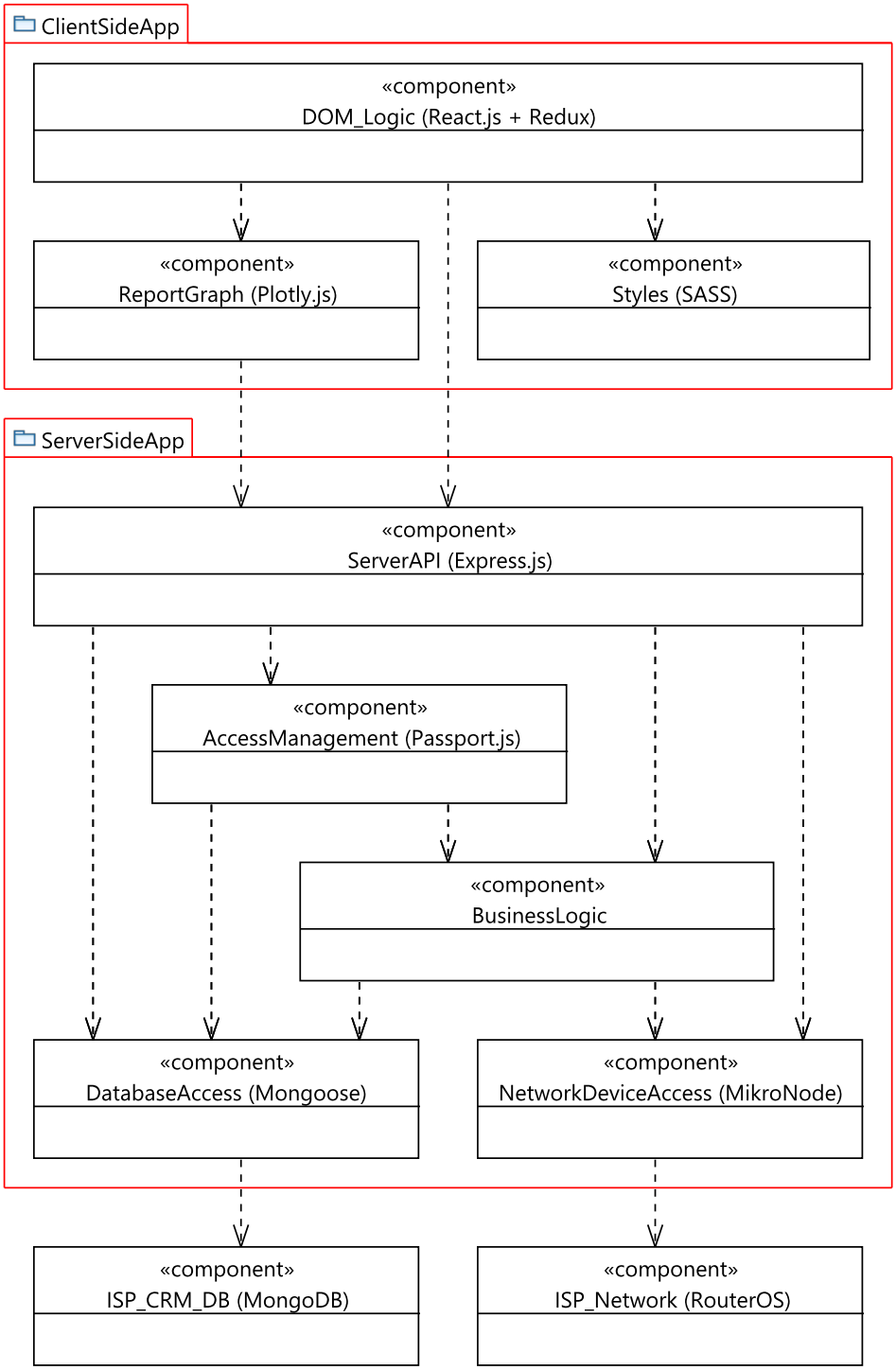
ServerSideApp–RouterOS Komunikační kanál mezi serverovou částí aplikace a RouterOS bude probíhat přes SSH. Balíček pro tuto komunikaci je součástí RouterOS ve výchozím nastavení.

ClientSideApp–ServerSideApp Komunikační kanál mezi klientskou a serverovou aplikací bude šifrovaný. Bude po něm probíhat přenos dat převážně ve formátu JSON.

WebBrowser–ClientSideApp Webový prohlížeč naváže spojení za použití ECC. Samotná komunikace pak bude šifrována pomocí AES. Komunikační protokol bude HTTPS nebo WSS.

4.2.2 Diagram komponent

V diagramu znázorněna komunikace mezi hlavními komponentami, ze kterých se systém skládá. Každá komponenta má zodpovědnost za chování určité části ve své aplikaci. Na obrázku 4.5 je zobrazena závislost komponent CRM systému.



Obrázek 4.5: Diagram komponent CRM systému

4.2.2.1 Komponenta ISP Network

Komponenta ISP_Network poskytuje rozhraní pro přístup do OS na síťovém zařízení (RouterOS). Tato komponenta je jediná z diagramu komponent, která není součástí systému. Je zodpovědností administrátorů sítě, aby jejich síťová zařízení byla nastavena takovým způsobem, že CRM systém bude moci s těmito zařízeními komunikovat, provádět v nich potřebné nastavení a získávat potřebná data.

4.2.2.2 Komponenta ISP CRM DB

Databáze označená jako ISP_CRM_DB obsahuje uložená data CRM systému. Spolu s databázovým systémem MongoDB tvoří komponentu pro práci s perzistentními daty. MongoDB bude s daty, jejich vazbami a formou uložení manipulovat na základě příchozích požadavků z CRM systému.

4.2.2.3 Aplikace ServerSideApp

Aplikace serverové části CRM systému se z důvodu větší složitosti dělí na několik komponent.

NetworkDeviceAccess Poskytuje funkce pro přístup k požadovaným datům a manipulaci s nastavením v síťových zařízení. Pro přístup do RouterOS poslouží knihovna MikroNode.

DatabaseAccess Poskytuje funkce k manipulaci se záznamy v databázi. Pro přístup k datům v MongoDB bude použita knihovna Mongoose.

BusinessLogic Obsahuje vnitřní byznys logiku serverové části aplikace. Tato komponenta tvoří jádro aplikace se zodpovědností za správnou transformaci dat, hlídání stavů síťových zařízení, periodické kontroly plateb a mnoho dalších procesů.

AccessManagement Slouží k autentizaci a autorizaci uživatelů. Chrání před neautorizovaným přístupem k částem určeným pouze omezené skupině uživatelů, spravuje hesla, ověřuje platnost sezení, řídí proces přihlašování a odhlašování uživatelů. K těmto účelům může sloužit knihovna Passport.js.

ServerAPI API serverové části vytvořené s pomocí knihovny Express.js bude poskytovat rozhraní pro přístup ostatním aplikacím.

4.2.2.4 Aplikace ClientSideApp

Aplikace vizualizační části CRM systému se pro snazší implementaci také dělí na více komponent.

ReportGraph Komponenta má na starosti transformaci získaných dat do grafů. Z nich bude snazší získávat informace o síti než z tabulek. V grafech budou například měřené propustnosti a odezvy spojů. K jednodušší vizualizaci bude sloužit knihovna Plotly.js.

Styles Kaskádové styly v jazyce SASS budou oddělené od logiky klientské aplikace, aby bylo možné snadno provést změnu v GUI.

DOM_Logic Tato komponenta bude manipulovat sestavovat DOM a tvořit tak UI ve webovém prohlížeči. Založená bude na React.js a v případě potřeby i na Redux. Data získá ze serverové části systému a převede je do vhodné reprezentace pro uživatele.

4.3 Objektový datový model

Objektový datový model znázorněný na obrázku 4.6 ukazuje, co bude v databázi uloženo a jakým způsobem provázáno. Vychází z analytického doménového modelu. Data budou uchovávána v kolekcích a budou vzájemně provázaná. Dále jsou popsány jednotlivé kolekce s jejich atributy a vazbami. Kromě uvedených atributů v modelu bude u každého záznamu atribut pro uchovávání textových poznámek.

4.3.1 Umístění

Informace o umístění skutečných objektů budou uloženy v kolekci **Location**.

Street (String) Ulice

Number (String) Číslo popisné / číslo orientační

Town (String) Název města či vesnice

PostalCode (String) Poštovní směrovací číslo

GPS_N (Double) Geografická souřadnice poledníku východní polokoule

GPS_E (Double) Geografická souřadnice rovnoběžky severní polokoule

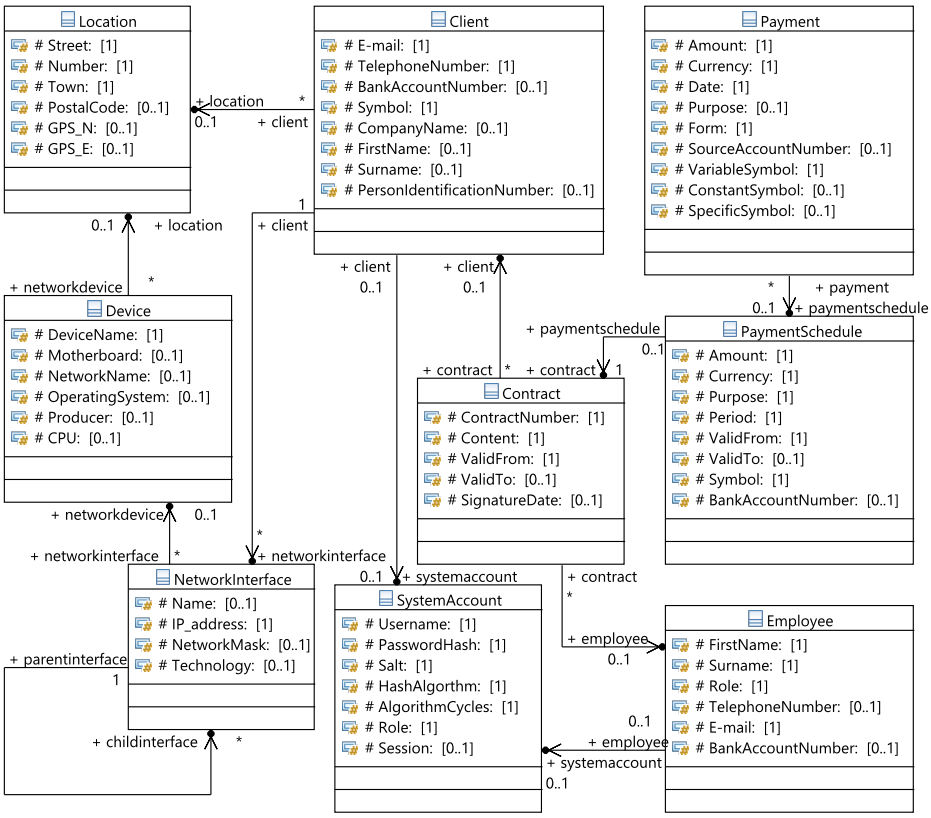
4.3.2 Zařízení

Informace o fyzických zařízení v síti budou uloženy v kolekci **Device**.

DeviceName (String) Oficiální označení zařízení od výrobce

Motherboard (String) Název základní desky

NetworkName (String) Název zařízení v síti



Obrázek 4.6: Objektový datový model

OperatingSystem (String) Operační systém běžící na zařízení

Producer (String) Výrobce zařízení

CPU (String) Název centrální výpočetní jednotky

4.3.3 Síťové rozhraní

Informace o síťových rozhraních a spojích budou uloženy v kolekci **NetworkInterface**.

Name (String) Uživatelský název rozhraní

IP_address (String) IP adresa nastavená na rozhraní

NetworkMask (String) Maska sítě

Technology (String) Technologie spoje

4.3.4 Klient

Informace o klientech poskytovatele budou uloženy v kolekci **Client**.

E-mail (String) E-mail na kontaktní osobu

TelephoneNumber (String) Telefonní číslo na kontaktní osobu

BankAccountNumber (String) Číslo bankovního účtu, ze kterého budou platby za služby odesílány (evidováno ke snadnější identifikaci plateb)

Symbol (Integer) Identifikační údaj klienta, kterým se prokazuje

CompanyName (String) Pokud je odběratelem právnická osoba, eviduje se její název

FirstName (String) Křestní jméno kontaktní osoby

Surname (String) Příjmení kontaktní osoby

PersonIdentificationNumber (Integer) Identifikační číslo osoby

4.3.5 Zaměstnanec

Informace o zaměstnancích u ISP budou uloženy v kolekci **Employee**.

FirstName (String) Křestní jméno

Surname (String) Příjmení

Role (String) Role u ISP

TelephoneNumber (String) Telefonní kontakt

E-mail (String) E-mailový kontakt

BankAccountNumber (String) Číslo bankovního účtu

4.3.6 Smlouva

Informace o smlouvách s klienty o poskytování služeb budou uloženy v kolekci **Contract**.

ContractNumber (Integer) Číslo obchodní smlouvy

Content (String) Obsah smlouvy s informacemi a podmínkách a poskytovaných službách

ValidFrom (Date) Datum nabytí platnosti smlouvy

ValidTo (Date) Datum, ke kterému je sjednaný konec platnosti smlouvy (vyplněno pouze u smluv na dobu určitou)

SignatureDate (Date) Datum podpisu smlouvy

4.3.7 Platební plán

Informace o platebních plánech budou uloženy v kolekci **PaymentSchedule**.

Amount (Decimal) Sjednaná částka za služby

Currency (String) Měna částky

Purpose (String) Popis služeb, ke kterým se plán vztahuje

Period (String) Možné periody plateb (NONE, MONTH, QUARTER, YEAR)

ValidFrom (Date) Datum, ke kterému je vztaženo první vyúčtování

ValidTo (Date) Datum, ke kterému bude vztaženo poslední vyúčtování

Symbol (Integer) Identifikační symbol definující plán

BankAccountNumber (String) Číslo účtu, na který mají platby přicházet

4.3.8 Platba

Informace o požadovaných a přijatých platbách budou uloženy v kolekci **Payment**.

Amount (Decimal) Velikost částky

Currency (String) Měna částky

Date (Date) Datum vystavení, nebo přijetí platby

Purpose (String) Účel vystavené platby, nebo popis příchozí platby

Form (String) Forma přijaté platby (PLAN – požadovaná, CASH – přijatá hotovost, BANK – příchozí bankovní)

SourceAccountNumber (String) Číslo účtu, z něhož byla platba přijata v případě bezhotovostní formy

VariableSymbol (Integer) Variabilní symbol přijaté platby

ConstantSymbol (Integer) Konstantní symbol přijaté platby

SpecificSymbol (Integer) Specifický symbol přijaté platby

4.3.9 Systémový účet

Informace o účtu sloužící k autentizaci a autorizaci uživatelů v CRM systému budou uloženy v kolekci **SystemAccount**.

Username (String) Uživatelské jméno

PasswordHash (String) Hash hesla

Salt (String) Kryptografická sůl

HashAlgorithm (String) Použitý hashovací algoritmus

AlgorithmCycles (Integer) Počet cyklů algoritmu

Role (String) Uživatelská role v systému

Session (String) Relace přihlášeného uživatele

4.3.10 Vazby

V této části jsou popsány vazby mezi záznamy. Pomocí nich lze tvořit rozšířené dotazy přes více kolekcí.

Device–Location Fyzické umístění zařízení

NetworkInterface–Device Síťové rozhraní zařízení

NetworkInterface–NetworkInterface Přímý komunikační spoj mezi síťovými rozhraními

Client–Location Umístění fyzického sídla klienta

Client–NetworkInterface Koncové rozhraní, ke kterému je klient připojen

Contract–Client Klient na smlouvě

Contract–Emplyee Zaměstnanec zodpovědný za smlouvu

PaymentSchedule–Contract Platební plán smlouvy

Payment–PaymentSchedule Platební plán patřící k platbě

Employee–SystemAccount Systémový účet zaměstnance

Client–SystemAccount Systémový účet klienta

4.4 Aplikační programové rozhraní

Na straně serveru bude aplikační programové rozhraní (API) vycházet z datového modelu a požadavků na systém. Některé metody budou přijímat další parametry, které budou sloužit zejména pro filtrování při selekci dat. V případě použití metod pro vytváření, úpravu a mazání dat v systému bude použit WS (WebSocket) k zaslání oznámení klientům, že došlo ke změně dat. Zároveň bude WS použit pro rozhraní vyžadující obousměrnou komunikaci. U všech požadavků od přihlášených uživatelů bude v cookie zasíláno **SessionID**. Popis jednotlivých rozhraní bude formátován tak, že na první pozici bude typ protokolu, za ním se může nacházet metoda a na následující pozici je cesta ke zdroji.

4.4.1 Analýza a nastavení spojů

WSS /tool:ping Kanál pro měření odezvy mezi dvěma zařízeními

- **SourceIP** IP adresa zdrojového rozhraní
- **SourceUsername** Uživatelské jméno zdrojového zařízení
- **SourcePassword** Heslo do zdrojového zařízení
- **DestinationIP** Cílová IP adresa rozhraní
- **Interval** Perioda zasílání paketů [ms]
- **Count** Počet opakování zasílání paketů

WSS /tool:throughput Kanál pro měření propustnosti mezi dvěma spoji

- **SourceIP** IP adresa zdrojového rozhraní
- **SourceUsername** Uživatelské jméno zdrojového zařízení
- **SourcePassword** Heslo do zdrojového zařízení
- **DestionationIP** IP adresa cílového rozhraní
- **DestionationUsername** Uživatelské jméno cílového zařízení
- **DestionationPassword** Heslo do cílového zařízení
- **ConnectionCount** Počet paralelních spojení
- **Duration** Doba běžícího testu [s]
- **Direction** Směr testu (SEND/RECEIVE/BOTH)
- **Interval** Perioda vracení aktuálních výsledků [s]
- **Protocol** Protokol (TCP/UDP)

WSS /tool:traceroute Kanál pro sledování celé trasy mezi dvěma rozhraními

SourceIP IP adresa zdrojového rozhraní

SourceUsername Uživatelské jméno zdrojového zařízení

SourcePassword Heslo do zdrojového zařízení

DestinationIP Cílová IP adresa rozhraní

Duration Doba trasování [s]

MaxHops Maximální počet skoků

HTTPS POST /setting/limit Nastavení maximální datové propustnosti IP adresy nebo podsítě, odpověď bude obsahovat informaci o úspěchu akce

SourceIP IP adresa rozhraní, na kterém má být limit nastaven

SourceUsername Uživatelské jméno zdrojového zařízení

SourcePassword Heslo do zdrojového zařízení

Identificator Zvolený identifikátor omezení

UploadLimit Omezení uploadu [b/s]

DownloadLimit Omezení downloadu [b/s]

TargetIP IP adresa nebo podsíť, pro kterou má být omezení nastaveno

HTTPS GET /setting/limit Získání seznamu omezení na požadovaném rozhraní

SourceIP IP adresa rozhraní

SourceUsername Uživatelské jméno zdrojového zařízení

SourcePassword Heslo do zdrojového zařízení

4.4.2 Čtení záznamů

Rozhraní pro čtení dat budou přijímat parametry vycházející z datového modelu systému. Každý parametr rozhraní se může vyskytovat v různých formátech. Pokud bude požadováno, aby byly vyhledány pouze záznamy s přesně zadanou hodnotou v parametru, bude parametr označen jako EQ. Pokud bude požadavek na vrácení záznamů s hodnotou větší než zadanou, bude parametr označen jako GTE (pro opačný případ označení LTE). Pokud bude požadavek na vyhledání záznamů, které obsahují zadaný řetězec, bude parametr označen jako IN[] (tento parametr se může vyskytovat vícekrát, proto je definovaný jako pole). Ke každé metodě pro čtení budou definovány v závorce formáty parametrů odděleny znakem plus.

HTTPS GET /location Získání záznamů o umístění objektů s parametry: Street (EQ + IN[]), Number (EQ + IN[]), Town (EQ + IN[]), PostalCode (EQ + IN[]), GPS_N (EQ + GTE + LTE), GPS_E (EQ + GTE + LTE) a Note (IN[])

HTTPS GET /location/{ID} Získání konkrétního záznamu o umístění se zadaným identifikátorem

HTTPS GET /device Získání záznamů o zařízeních s parametry: DeviceName (EQ + IN[]), Motherboard (EQ + IN[]), NetworkName (EQ + IN[]), OperatingSystem (EQ + IN[]), Producer (EQ + IN[]), CPU (EQ + IN[]), Note (IN[]) a LocationID (EQ)

HTTPS GET /device/{ID} Získání konkrétního záznamu o zařízení se zadaným identifikátorem

HTTPS GET /networkinterface Získání záznamů o síťových rozhraních s parametry: Name (EQ + IN[]), IP_address (EQ + IN[]), Technology (EQ + IN[]), NetworkMask (EQ + IN[]), Note (IN[]), DevideID (EQ), NetworkInterfaceParentID (EQ) a ClientID (EQ)

HTTPS GET /networkinterface/{ID} Získání záznamu o konkrétním síťovém rozhraní se zadaným identifikátorem

HTTPS GET /client Získání záznamů o klientech s parametry: E-mail (EQ + IN[]), TelephoneNumber (EQ + IN[]), BankAccountNumber (EQ + IN[]), Symbol (EQ + GTE + LTE), CompanyName (EQ + IN[]), FirstName (EQ + IN[]), Surname (EQ + IN[]), PersonalIdentificationNumber (EQ + GTE + LTE), Note (IN[]), LocationID (EQ) a SystemAccountID (EQ)

HTTPS GET /client/{ID} Získání konkrétního záznamu o klientovi se zadaným identifikátorem

HTTPS GET /employee Získání záznamů o zaměstnancích s parametry: FirstName (EQ + IN[]), Surname (EQ + IN[]), Role (EQ + IN[]), TelephoneNumber (EQ + IN[]), E-mail (EQ + IN[]), Note (IN[]), BankAccountNumber (EQ + IN[]) a SystemAccountID (EQ)

HTTPS GET /employee/{ID} Získání konkrétního záznamu o zaměstnanci se zadaným identifikátorem

HTTPS GET /contract Získání záznamů o smlouvách s parametry: ContractNumber (EQ + GTE + LTE), Content (IN[]), ValidFrom (GTE + LTE), ValidTo (GTE + LTE), SignatureDate (GTE + LTE), Note (IN[]), ClientID (EQ) a EmployeeID (EQ)

HTTPS GET /contract/{ID} Získání konkrétního záznamu o smlouvě se zadaným identifikátorem

HTTPS GET /paymentschedule Získání záznamů o platebních plánech s parametry: Amount (EQ + GTE + LTE), Currency (EQ), Purpose (EQ + IN[]), Period (EQ), ValidFrom (GTE + LTE), ValidTo (GTE +

LTE), Symbol (EQ + GTE + LTE), BankAccountNumber (EQ + IN[]), Note (IN[]) a ContractID (EQ)

HTTPS GET /paymentschedule/{ID} Získání konkrétního záznamu o platebním plánu se zadaným identifikátorem

HTTPS GET /payment Získání záznamů o platbách s parametry: Date (GTE + LTE), Amount (EQ + GTE + LTE), Currency (EQ), Purpose (EQ + IN[]), Form (EQ), SourceAccountNumber (EQ + IN[]), VariableSymbol (EQ + GTE + LTE), ConstantSymbol (EQ + GTE + LTE), SpecificSymbol (EQ + GTE + LTE), Note (IN[]) a PaymentScheduleID (EQ)

HTTPS GET /payment/{ID} Získání konkrétního záznamu o platbě se zadaným identifikátorem

4.4.3 Vytvoření záznamů

Rozhraní pro vytváření záznamů budou přijímat parametry vycházející z datového modelu.

HTTPS POST /location Vytvoření záznamu o umístění s parametry: street (String), number (String), town (String), postalCode (String), gpsN (Double) a gpsE (Double)

HTTPS POST /device Vytvoření záznamu o zařízení s parametry: deviceName (String), motherboard (String), networkName (String), operatingSystem (String), producer (String), cpu (String) a locationID

HTTPS POST /networkinterface Vytvoření záznamu o síťovém rozhraní s parametry: name (String), ipAddress (String), technology (String), networkMask (String), parentInterfaceID, deviceID a clientID

HTTPS POST /client Vytvoření záznamu o klientovi s parametry: email (String), telephoneNumber (String), symbol (Integer), bankAccountNumber (String), companyName (String), firstName (String), surname (String), personalIdentificationNumber (String), locationID (String) a systemAccountID

HTTPS POST /employee Vytvoření záznamu o zaměstnanci s parametry: firstName (String), surname (String), role (String), telephoneNumber (String), email (String), bankAccountNumber (String) a systemAccountID

HTTPS POST /contract Vytvoření záznamu o smlouvě s parametry: contractNumber (Integer), content (String), validFrom (Date), validTo (Date), signatureDate (Date), employeeID a clientID

HTTPS POST /paymentschedule Vytvoření záznamu o platebním plánu s parametry: amount (Decimal), currency (String), purpose (String), period (NONE / MONTH / QUARTER / YEAR), validFrom (Date), validTo (Date), symbol (Integer), bankAccountNumber (String) a contractID

HTTPS POST /payment Vytvoření záznamu o platbě s parametry: date (Date), amount (Decimal), currency (String), curpose (String), form (PLAN / CASH / BANK), sourceAccountNumber (String), variableSymbol (Integer), constantSymbol (Integer), specificSymbol (Integer) a paymentScheduleID

4.4.4 Aktualizace záznamů

Rozhraní pro změnu hodnot v záznamech budou přijímat parametry vycházející z datového modelu systému. Každá změna upravuje hodnoty záznamu se zadaným ID.

HTTPS PUT /location/{ID} Změna záznamu o umístění v parametrech: street (String), number (String), town (String), postalCode (String), gpsN (Double) a gpsE (Double)

HTTPS PUT /device/{ID} Změna záznamu o zařízení v parametrech: deviceName (String), motherboard (String), networkName (String), operatingSystem (String), producer (String), cpu (String) a locationID

HTTPS PUT /networkinterface/{ID} Změna záznamu o síťovém rozhraní v parametrech: name (String), ipAddress (String), technology (String), networkMask (String), parentInterfaceID, deviceID a clientID

HTTPS PUT /client/{ID} Změna záznamu o klientovi v parametrech: email (String), telephoneNumber (String), symbol (Integer), bankAccountNumber (String), companyName (String), firstName (String), surname (String), personalIdentificationNumber (String), locationID a systemAccountID

HTTPS PUT /employee/{ID} Změna záznamu o zaměstnanci v parametrech: firstName (String), surname (String), role (String), telephoneNumber (String), email (String), bankAccountNumber (String) a systemAccountID

HTTPS PUT /contract/{ID} Změna záznamu o smlouvě v parametrech: contractNumber (Integer), content (String), validFrom (Date), validTo (Date), signatureDate (Date), paymentScheduleID, employeeID a clientID

HTTPS PUT /paymentschedule/{ID} Změna záznamu o platebním plánu v parametrech: amount (Decimal), currency (String), purpose

(String), period (NONE / MONTH / QUARTER / YEAR), validFrom (Date), validTo (Date), symbol (Integer), bankAccountNumber (String) a contractID

HTTPS PUT /payment/{ID} Změna záznamu o platbě v parametrech: date (Date), amount (Decimal), currency (String), curpose (String), form (PLAN / CASH / BANK), sourceAccountNumber (String), variableSymbol (Integer), constantSymbol (Integer), specificSymbol (Integer) a paymentScheduleID

4.4.5 Odstranění záznamů

Rozhraní pro odstranění záznamů má jediný parametr, který identifikuje konkrétní záznam. Systém povolí záznam odstranit pouze v případě, že záznam není provázaný s jiným záznamem.

HTTPS DELETE /location/{ID} Odstranění záznamu o umístění se zadaným identifikátorem

HTTPS DELETE /device/{ID} Odstranění záznamu o zařízení se zadaným identifikátorem

HTTPS DELETE /networkinterface/{ID} Odstranění záznamu o síťovém rozhraní se zadaným identifikátorem

HTTPS DELETE /client/{ID} Odstranění záznamu o klientovi se zadaným identifikátorem

HTTPS DELETE /employee/{ID} Odstranění záznamu o zaměstnanci se zadaným identifikátorem

HTTPS DELETE /contract/{ID} Odstranění záznamu o smlouvě se zadaným identifikátorem

HTTPS DELETE /paymentschedule/{ID} Odstranění záznamu o platebním plánu se zadaným identifikátorem

HTTPS DELETE /payment/{ID} Odstranění záznamu o platbě se zadaným identifikátorem

4.4.6 Řízení uživatelských účtů

HTTPS POST /login/client Přihlášení klienta do systému a získání SessionID (rozhraní dostupné nepřihlášeným uživatelům)

Type Typ přihlašovacího údaje (EMAIL / TELEPHONE / USERNAME)

Username Unikátní údaj klienta

Password Heslo

HTTPS POST /login/employee Přihlášení zaměstnance do systému a získání SessionID (rozhraní dostupné nepřihlášeným uživatelům)

Type Typ přihlašovacího údaje (EMAIL / TELEPHONE / USERNAME)

Username Unikátní údaj zaměstnance

Password Heslo

HTTPS POST /account Vytvoření uživatelského účtu

Username Uživatelské jméno

Password Heslo

HashAlgorithm Hashovací algoritmus

Role Role účtu v systému

Count Počet cyklů algoritmu

HTTPS POST /logout Odhlášení uživatele ze systému a odstranění jeho SessionID

HTTPS GET /account/{ID} Získání konkrétního záznamu o systémovém účtu se zadaným ID

HTTPS POST /account/ID/newpassword Vygenerování nového hesla do účtu s konkrétním ID

HTTPS PUT /own/password Změna hesla v účtu

OldPassword Původní heslo

NewPassword Nové heslo

HTTPS DELETE /account/{ID} Odstranění záznamu o systémovém účtu se zadaným identifikátorem

4.4.7 Hlídání změn v datech

Jelikož může data měnit více přihlášených uživatelů a klientská aplikace může mít načtená nějaká data, bude existovat rozhraní pro navázání komunikačního kanálu, přes který mohou klienti získávat informace o změnách.

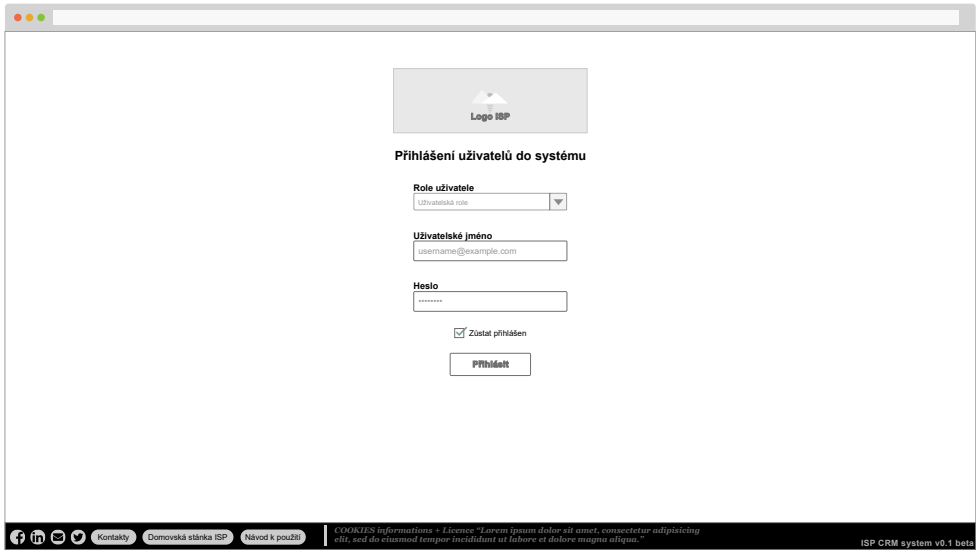
WSS /changelog Kanál pro zasílání informací, ve které kolekci a v jakém atributu došlo ke změně dat, aby klientská aplikace v případě, že má načtená data z dané kolekce, si je zaktualizovala, data budou obsahovat pouze názvy kolekcí a atributů, ve kterých došlo ke změně. Změny v SessionID nebudou propagovány, k nim nebude mít přístup žádný uživatel z důvodu zajištění bezpečnosti.

4.5 Uživatelské rozhraní systému

Uživatelské rozhraní klientské části systému bude vycházet z WireFrame (WF), které budou představeny v této části. Hlavní obrazovky UI jsou pro přihlášení, úvodní stránky uživatelů, vytváření záznamů, úpravu záznamů a analýza sítě.

4.5.1 Přihlášení uživatele

Přihlašovací obrazovka pro uživatele je na obrázku 4.7. V horní části bude logo ISP, následuje výběr role, pod jakou se chce uživatel přihlásit. Pro usnadnění použitelnosti klientům bude výchozí hodnotou role klient, administrátor sítě musí hodnotu změnit, aby se přihlásil do správcovského režimu. Následuje pole pro uživatelské jméno, kam lze zadat emailovou adresu nebo zvolené uživatelské jméno. Následuje pole pro heslo a pomocí zaškrtávacího pole lze nastavit, aby uživatel nebyl automaticky odhlášen při dlouhé nečinnosti. Samotné přihlášení se provede tlačítkem **Přihlásit**, které zajistí odeslání zadaných údajů po zabezpečeném kanále na server, na němž se zadané hodnoty ověří a v případě platnosti bude uživatel puštěn do systému.



Obrázek 4.7: Přihlašovací obrazovka

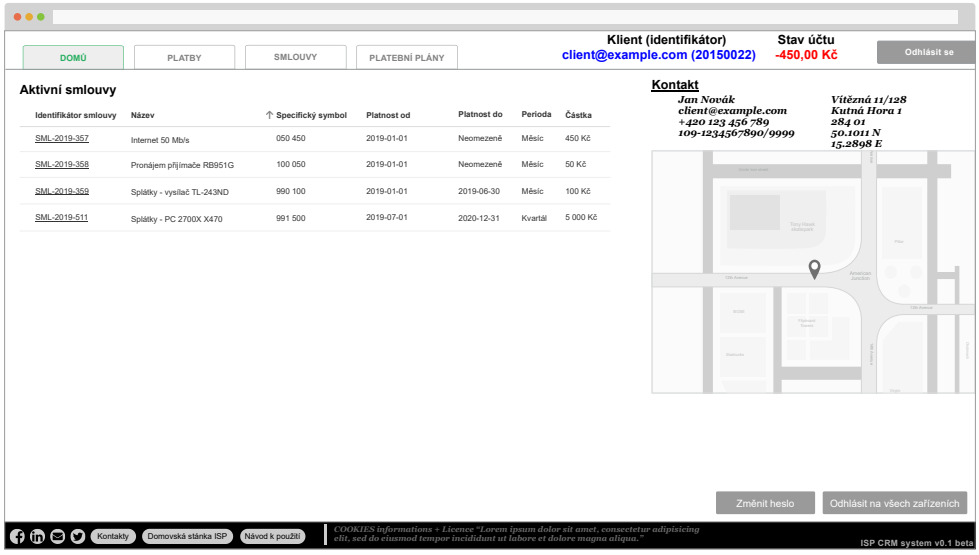
4.5.2 Klientská část

Úvodní obrazovka pro přihlášeného klienta je na obrázku 4.8. V horní části obrazovky se nachází menu pro přístup k různým záznamům, údaje o emailové adrese s identifikátorem přihlášeného uživatele, aktuální stav účtu a tlačítko pro odhlášení, které zajistí odebrání SessionID v jeho prohlížeči. Menu bude

zobrazené na všech obrazovkách přihlášeného klienta a vyznačena bude aktuálně zobrazená položka.

Na hlavní stránce bude seznam aktivních smluv klienta. Po kliknutí na identifikátor se uživatel dostane k zobrazení detailu smlouvy. Na pravé straně bude zobrazen kontakt na klienta. Ten si může klient sám upravit po kliknutí na nadpis. Pod kontaktem je zobrazená mapa s vyznačením polohy sídla klienta.

V pravém dolním rohu je tlačítko pro změnu hesla do systému a tlačítko pro odhlášení na všech zařízeních, což bude zajištěno tak, že se na straně serveru odstraní SessionID, čímž se zneplatní uložené SessionID ve všech prohlížečích, kde byl uživatel přihlášen.

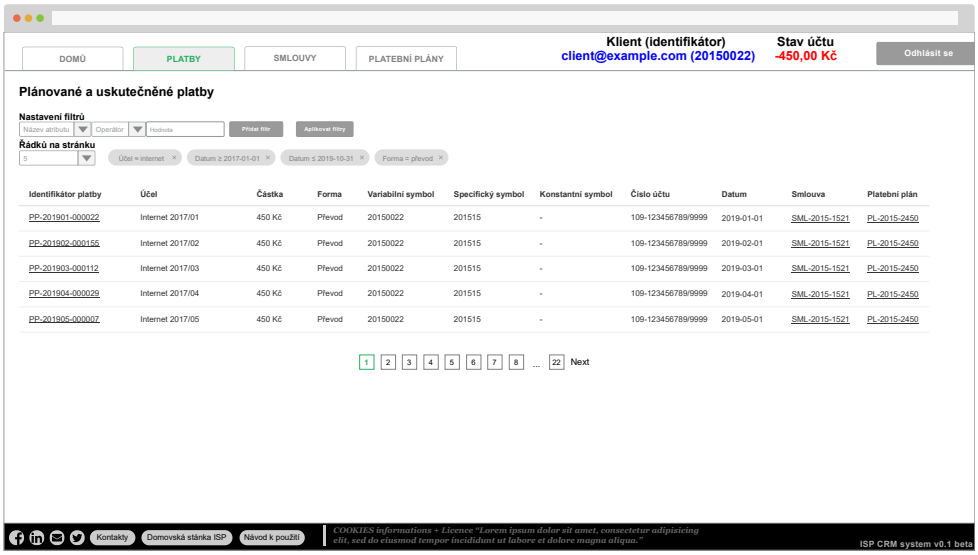


Obrázek 4.8: Hlavní obrazovka klienta

4.5.2.1 Platby, smlouvy a platební plány

Každý klient si bude moci zobrazit platby, smlouvy a platební plány, které patří jeho osobě. Všechny přehledy budou mít podobné rozložení jako je na obrázku 4.9. V horní části bude nadpis pro lepší orientaci uživatele, kde se právě v systému nachází. Pod nadpisem budou ovládací prvky pro filtrování obsahu. Uživatel postupně vybere název atributu, nad kterým chce vytvořit filtr, operátor a hodnotu. Každý atribut bude mít vlastní seznam operátorů vycházející z API pro čtení záznamů. Uživatel si bude moci nastavit více filtrů najednou, které budou zobrazeny pod tímto formulářem. Dále bude mít uživatel možnost volby počtu zobrazených záznamů na stránku. Seznam zvolených filtrů bude zobrazen nad tabulkou s obsahem. Každý filtr půjde zrušit kliknutím na jeho křížek. Při změně filtrů nebude zobrazován žádný obsah, až po kliknutí na tlačítko aplikovat filtry se zobrazí platný obsah pro

vybrané filtry v tabulce. Hlavička tabulky se záznamy se bude lišit podle toho, jaké záznamy si uživatel zobrazuje. Některé sloupce budou obsahovat odkazy pro zobrazení detailu záznamu. Pokud se do jedné tabulky nevejde veškerý obsah, budou pod ní tlačítka na přechod na další stránku.



Obrázek 4.9: Přehled záznamů u klienta

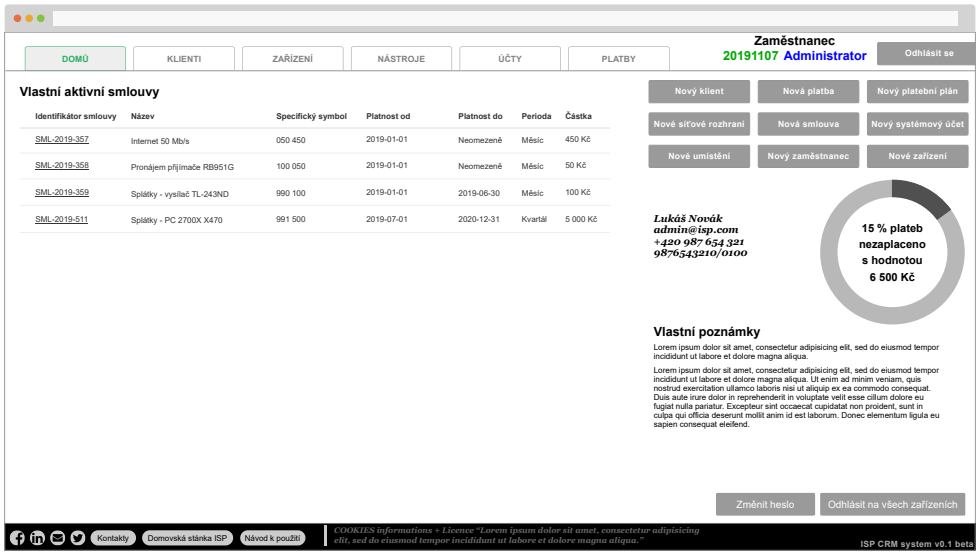
4.5.3 Administrátorská část

Hlavní menu správcovského uživatelského rozhraní v horní části bude obsahovat položky pro správu záznamů o klientech, zařízeních, účtech, platbách a nástroje pro měření stability síťových spojů, číslo zaměstnance a jeho role.

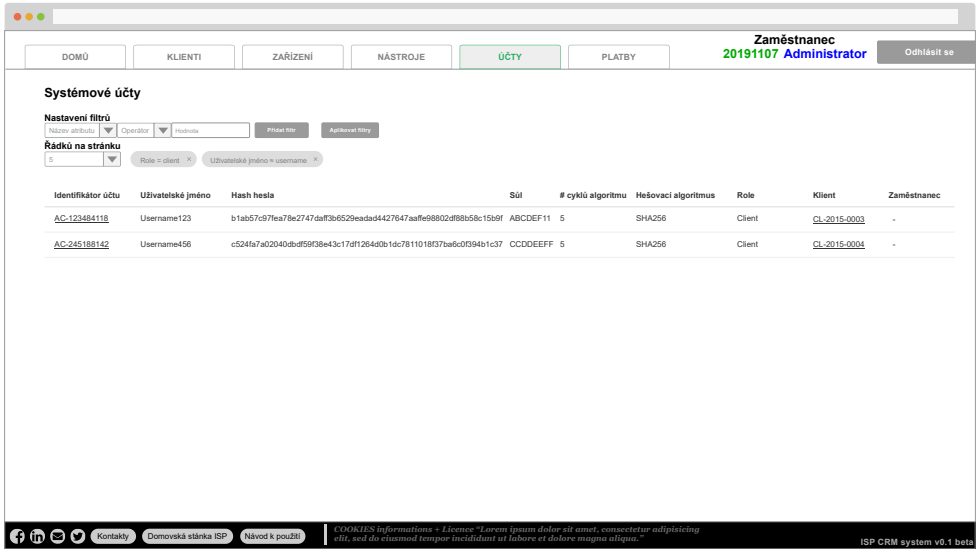
Úvodní obrazovka pro správce sítě na obrázku 4.10 se od klientské značně liší. Na hlavní stránce bude mít seznam svých aktivních smluv s klienty. V pravé části budou tlačítka na vytváření záznamů do systému, informace o přihlášeném zaměstnanci, graf s procentuálním podílem nesplacených plánovaných plateb a jejich celkovou hodnotou a poznámky ke svému účtu. Stejně jako u klienta budou v pravé spodní části tlačítka na změnu hesla do systému a odhlášení na všech zařízeních.

4.5.3.1 Přehled záznamů

Správce sítě si bude moci zobrazit veškeré záznamy v systému bez ohledu, zda je jejich vlastníkem. Filtrování záznamů, stránkování a zobrazení detailů přes identifikátory záznamů bude fungovat stejně jako u klienta. Na obrázku 4.11 je zobrazen seznam účtů v systému. Stejným způsobem bude mít možnost zobrazení klientů, plateb, platebních plánů, síťových rozhraní, smluv, umístění, zaměstnanců a zařízení.



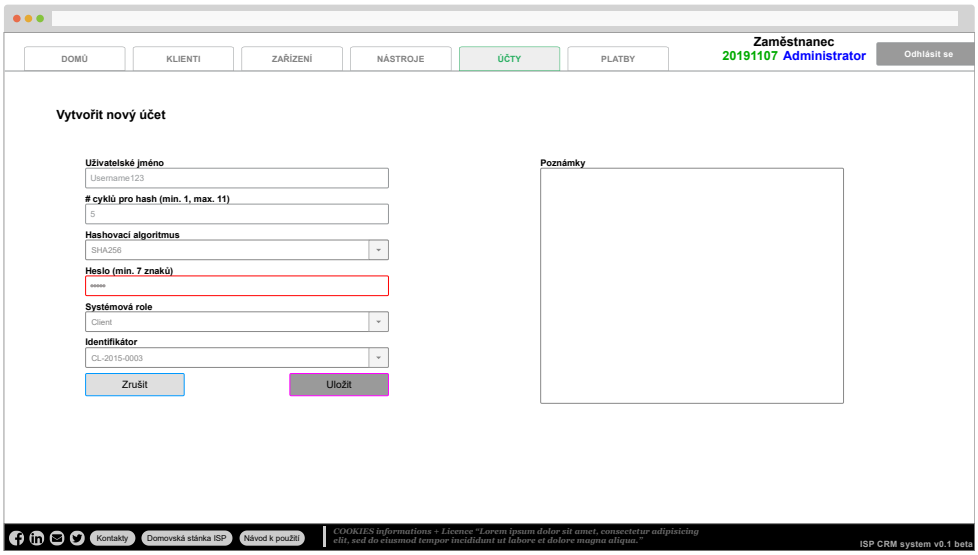
Obrázek 4.10: Hlavní obrazovka správce



Obrázek 4.11: Pohled na záznamy u správce

4.5.3.2 Správa záznamů

V administrátorském režimu lze přidávat, upravovat a mazat libovolný obsah. Pro přidání nového záznamu bude použit formulář, který bude podobný obrázku 4.12. Textová pole budou odpovídat atributům záznamu. Jako příklad je na obrázku tvorba nového účtu do systému. Některé položky bude možné generovat, pokud obsah v poli nebude odpovídat požadavkům, bude rámeček pole červený. Až teprve po kliknutí na tlačítko **Uložit** se záznam vloží do databáze a uživatel bude přesměrován na stránku pro úpravu záznamu.

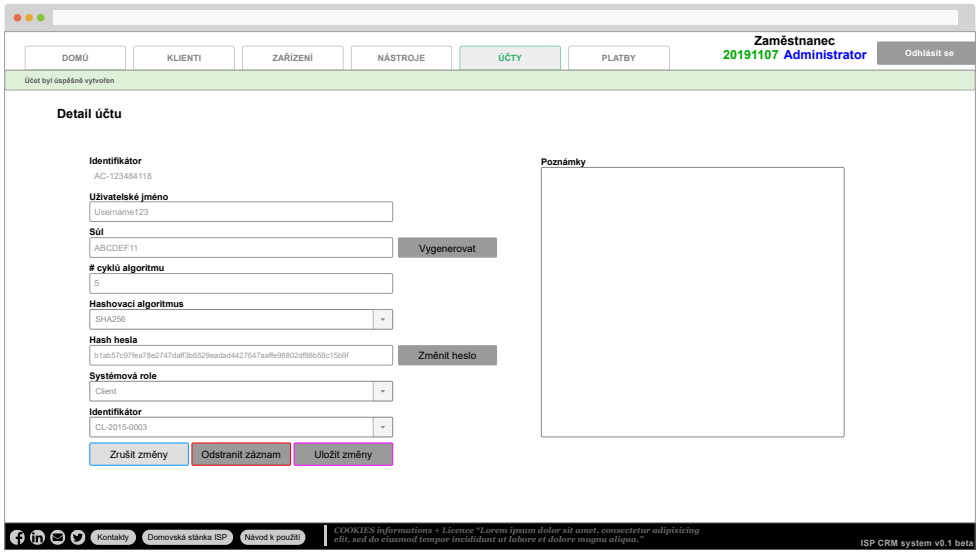


Obrázek 4.12: Přidání nového záznamu

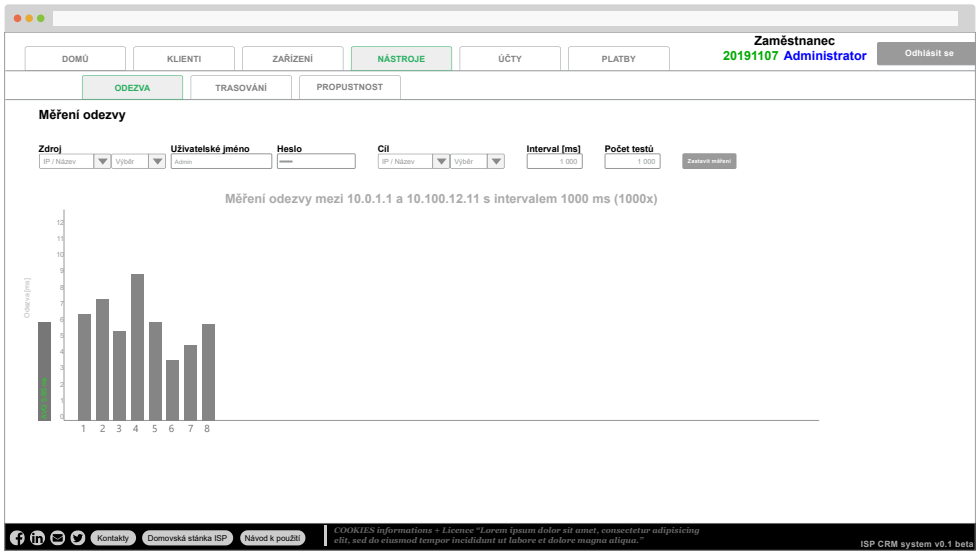
Pokud uživatel vybere nějaký konkrétní záznam, zobrazí se mu obrazovka podobná obrázku 4.13. Na ní bude moci uživatel upravovat obsah atributů, případně záznam odstranit či vrátit aktuální provedené změny zpět. Položky, které budou pouze ke čtení, budou zobrazeny stejným způsobem jako je na na obrázku 4.13 zobrazen identifikátor. Stejné okno bude k dispozici i pro klienty.

4.5.3.3 Administrátorské nástroje

Obrazovka pro měření síťových spojů bude ve třech verzích. Měření odezvy spojů (na obrázku 4.14), trasování a měření datové propustnosti. Parametry potřebné pro měření se budou zadávat ručně v horní části obrazovky. Vycházet budou z definice API. Měření odezvy bude ve sloupcovém grafu stejně jako měření propustnosti. Vedle grafu bude zobrazena průměrná hodnota. Trasování bude vypisováno pouze v textovém formátu.



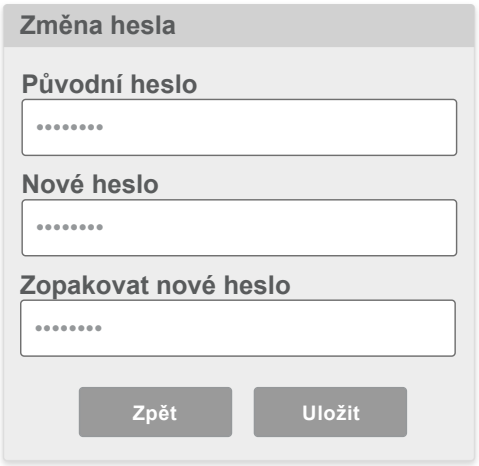
Obrázek 4.13: Zobrazení konkrétního záznamu



Obrázek 4.14: Nástroje pro správce

4.5.4 Změna hesla

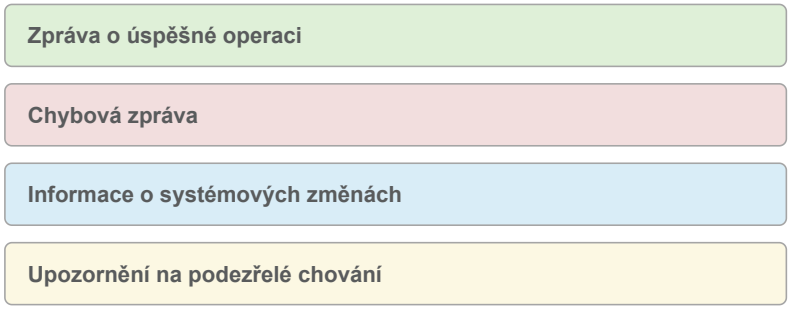
Formulář pro změnu hesla bude ve formě vyskakovacího okna. Jeho podoba je zobrazena na obrázku 4.15. Uživatel bude vyzván k zadání nového hesla a jeho potvrzení. Po uložení bude v systému vygenerován jeho hash a uložen do databáze.



Obrázek 4.15: Změna hesla

4.5.5 Systémové hlášky

Pokud bude ze strany systému potřeba oznámit uživateli nějakou informaci, bude to řešeno pomocí hlášky v horní části okna. Budou existovat čtyři typy těchto oznámení. Jejich grafická podoba je na obrázku 4.16. Zprávy o úspěšné operaci budou mít zelený podklad, chybové zprávy červený podklad, informace o systémových změnách modrý podklad a upozornění na podezřelé chování žlutý podklad.



Obrázek 4.16: Systémové hlášky

Kapitola 5

Implementace

Implementace systému je postavena tak, aby byly použity součásti ve verzích s dlouhodobou podporou označující se jako LTS (Long Term Support), pokud to bude možné z hlediska potřebných funkcí a jejich dostupnosti. Tím bude zajištěna dlouhodobá podpora celého systému a nebude náchylný na brzký konec podpory použitých částí. To pomůže snížit náklady na údržbu systému díky delší periodě nutné údržby.

5.1 Podpůrné komponenty a nástroje pro vývoj

Ještě před samotným vývojem aplikace je potřeba nakonfigurovat nástroje pro její vývoj. Většina nastavení bude v konfiguračním souboru pro Webpack a NodeJS. V konfiguraci pro Webpack bude popsáno, co a jak přeložit, provázat a vytvořit, aby vznikly potřebné výstupní soubory. Pro NodeJS bude nakonfigurováno, jaké balíčky a v jakých verzích je potřeba stáhnout pro běh každé části systému. Při implementaci byl použit NodeJS ve verzi 12 LTS.

5.1.1 Serverová a klientská část

Jelikož je systém rozdělen na více částí, je vhodné rozdělit implementaci do více adresářů. Proto bude projekt obsahovat adresář **server** a adresář **client**. Prvky v adresáři server nebude nutné překládat do nižších verzí ES a použit bude překlad do ES6. V adresáři client bude nutné pro běh aplikace i ve starších prohlížečích přeložit zdrojové soubory do ES5.

5.1.2 Postman

K testování aplikačního rozhraní serverové části bude použita aplikace Postman. Ta umožňuje pomocí různých metod volat API. Lze také vytvořit testovací sestavy a tím lépe odchytávat chyby již během vývoje. Podporuje i autentizaci pomocí různých protokolů.

5.1.3 MongoDB Compass

MongoDB Compass je grafický nástroj pro MongoDB. Umožňuje práci s obsahem pomocí CRUD operací a upravovat nastavení DB. Připojení k databázi lze provést pomocí nezabezpečeného kanálu, pomocí SSL/TLS i pomocí SSH. Tento nástroj umožňuje zobrazit náročnost dotazů, použití paměti a mnoho dalších parametrů o databázi. Lze v něm také nastavovat indexy, importovat a exportovat data. Veškeré vlastnosti jsou popsány v oficiální dokumentaci, která je dostupná na adrese `https://docs.mongodb.com/compass/current/`.

5.1.4 Microsoft Visual Studio Community

Pro vývoj softwaru použiji IDE Visual Studio Coommunity od společnosti Microsoft, které je možné použít zdarma v případě použití pro výuku, vědecký výzkum nebo pro přispívání do open source projektů. Pro jednotlivce jej lze použít zdarma i v případě komerční použití. [46]

Jeho výhoda spočívá ve velkém množství doplňků. Mimo jiné umožňuje zvýrazňovat kód, propojit projekt s verzovacím systémem Git, podporuje mnoho programovacích a skriptovacích jazyků včetně frameworků. Umožňuje spolupráci se správci balíčků a odchytávat chyby v kódu. Další výhodou ve veliké množství klávesových zkratek, které jsou popsány na webové adrese `https://visualstudio.microsoft.com/cs/vs/features/develop/`.

5.1.5 Nastavení pro Webpack a NodeJS

Systém je rozdělen na serverovou a klientskou část, proto je rozdělena i konfigurace do dvou částí. Klientská část má složitější konfiguraci pro Webpack z důvodu překladu kódu do starší verze ES. Serverová i klientská část má konfiguraci pro TS v souborech **tsconfig.json**, obsahující konfiguraci pro překlad z TS do ES, pro Webpack v souborech s předponou **webpack.config** s nastavením, jak má zacházet s různými typy souborů a pro NodeJS v souborech **package.json** s nastavením spouštěcích skriptů pro sestavení vývojových a produkčních verzí včetně seznamu balíčků potřebných pro běh každé aplikace.

5.2 Implementace komponent

Implementace je rozdělena na implementaci klientské a serverové části. Nejdůležitější je řádná implementace serverové části, která tvoří centrální aplikaci v systému. Spojuje veškeré části systému jako je klientská část s uživatelským rozhraním, databáze a síťová zařízení. Práce obsahuje implementaci serverové části a přípravu pro implementaci klientské části.

5.2.1 Použité knihovny

V systému je použita řada dalších knihoven, které nejsou popsány v části o výběru vhodných technologií. Při implementaci je dáván důraz na výběr knihoven s perspektivou do budoucna a otevřenou licencí.

mongoose Pro operace s daty v databázi je použita knihovna mongoose, která umožňuje připojení k MongoDB a provádět manipulaci s daty v kolekcích.

joi Pro validaci atributů zasílaných objektů je použita knihovna joi. Ta umožňuje pomocí předpřipravených funkcí validovat nejen základní typy, ale kontrolovat například i správnost formátu zadaného e-mailu, IP adres a mnoho dalších. Umožňuje také validovat hodnoty pomocí regulárních výrazů.

bcrypt a crypto Ke generování náhodné soli je použita knihovna bcrypt, hashování a generování nových hesel je zajištěno pomocí knihovny crypto.

socket.io Pro obousměrnou komunikaci je v systému použita knihovna socket.io. Ta umožňuje v případě dostupnosti protokolu WS u klienta jej použít, v případě nedostupnosti WS na straně klienta používá long-polling.

node-routeros Ke komunikaci se síťovými zařízeními s RouterOS je použita knihovna node-routeros. Ta poskytuje rozhraní pro snadnou manipulaci s RouterOS rozhraním a současně umožňuje vytvořit kanál pro poslouchání odpovědí systému v zařízení.

node-schedule K pravidelnému automatickému generování požadavků na platby podle platebního plánu je využit plánovač, který poskytuje knihovna node-schedule. V systému je plánovač využit ke generování platebních požadavků podle plánů. Tato funkce bude spouštěna denně po překročení půlnoci.

5.2.2 Strana serveru

Na straně serveru poběží aplikace pro obsluhu požadavků, které nelze vyřídit na straně klienta. Její rozhraní bude odpovídat API, které je popsané v kapitole s návrhem řešení. Rozhraní serveru bude komunikovat převážně pomocí protokolu HTTP a v odesílaných odpovědích budou používány běžné stavové kódy popsané na adrese `https://developer.mozilla.org/en-US/docs/Web/HTTP/Status`, kterým bude přidružená zpráva s detailnější odpovědí. Každý požadavek bude zpracován asynchronně, aby nezpomaloval hlavní smyčku v Node.js a server dokázal obsloužit více požadavků.

5.2.3 Zabezpečení přenášených dat po veřejném kanále

Jelikož musí být přenášená data utajená, budou šifrována. K tomu aplikace vyžaduje dva soubory. Jeden soubor s veřejným certifikátem a druhý s privátním klíčem. Při vývoji jsou použity klíče podepsané samy sebou, bez certifikační autority. Pro nasazení do produkčního prostředí musí administrátor tyto klíče vyměnit za vlastní, které budou podepsané certifikační autoritou. Pomocí těchto klíčů se vytvoří HTTPS a WSS server pro obsluhování požadavků.

5.2.4 Autentizace a autorizace

Do systému budou mít přístup pouze osoby s vytvořeným systémovým účtem. Při autentizaci uživatel získá vygenerované SessionID, pomocí kterého se bude při odeslání požadavku na server prokazovat. To bude v systému unikátní. Jelikož ne každý uživatel může provádět veškeré dostupné operace, bude mít ve svém účtu roli. Každé rozhraní bude mít definováno, jaké role mohou k rozhraní přistupovat. Pokud není žádná role u rozhraní definována, rozhraní je přístupné bez přihlášení.

5.2.5 Strana klienta

Strana klienta není v této práci implementována, přesto je k její implementaci vytvořena šablona a konfigurace Webpack a NodeJS pro její vývoj.

5.2.5.1 Šablony bootstrap

Pro usnadnění vývoje bude k usnadnění tvorby responzivního designu použita knihovna **Bootstrap**. Obsahuje mnoho připravených šablon, čímž urychluje vývoj uživatelského rozhraní webových aplikací. Její licence je MIT a je navržena pro spolupráci s HTML, SASS a JS.

5.2.5.2 Axios

Knihovna Axios slouží k asynchronnímu volání HTTP požadavků. Tím se zajistí, že klientská část aplikace nebude čekat na odpověď serveru a poběží dále. Po příchodu odpovědi ze serveru budou příchozí data zpracována libovolnou funkcí, kterou předáme při volání HTTP požadavku.

Kapitola 6

Vyhodnocení

V této kapitole je provedeno vyhodnocení nákladů na vývoj, provoz a údržbu softwaru. Dále obsahuje popis limitů tohoto řešení a popis možností budoucího rozvoje systému.

6.1 Finanční náklady na projekt

Pro vyhodnocení finančních nákladů byla použita aplikace **ProjectLibre**, který je určen k plánování projektů. Je pod licencí CPAL, která umožňuje oproti komerčním produktům volné užití tohoto softwaru, čímž šetří další náklady.

Vypočítané časové a finanční náklady jsou pouze hrubým odhadem. K výpočtu finančních nákladů na lidské zdroje byly použity průměrné hrubé měsíční mzdy k jednotlivým pozicím v České republice, které jsou dostupné na adrese `https://www.platy.cz/platy`. Hodinové náklady ke každé pozici byly vypočítány z průměrné hrubé měsíční mzdy navýšené o náklady zaměstnavatele na zdravotní a sociální pojištění (9 % a 25 %). Výsledek byl podělen 168, což je průměrný počet pracovních hodin v měsíci. Výsledek byl zaokrouhlen na desítky Kč nahoru. Rozpis jednotlivých hodinových nákladů je v tabulce 6.1.

Tabulka 6.1: Hodinové náklady na zaměstnance

Pracovní pozice	**Hodinový náklad**
IT analytik	400 Kč
SW architekt	670 Kč
Web designér	330 Kč
Programátor JavaScript	410 Kč
Tester	330 Kč
IT specialista	330 Kč

		Jméno	Trvání	Předchůdci	Jména zdrojů	Náklady
1		**ISP CRM system**	**52 dní**			**303040 Kč**
2		Zahájení	1 den		IT specialista	2640 Kč
3		**Analýza**	**51 dní**			**44800 Kč**
4		Sběr funkčních a nefunkčních požadavků	7 dní	2	IT Analytik	22400 Kč
5		Případy užití	3 dní	4	IT Analytik	9600 Kč
6		Doménový model	4 dní	4	IT Analytik	12800 Kč
7		**Návrh**	**39 dní**			**82800 Kč**
8		Technologie pro vývoj	5 dní	4	SW architekt	26800 Kč
9		Objektový datový model	2 dní	8	SW architekt	10720 Kč
10		Grafické uživatelské rozhraní	7 dní	8	Web designer	18480 Kč
11		Aplikační programové rozhraní	5 dní	8	SW architekt	26800 Kč
12		**Implementace**	**31 dní**			**127920 Kč**
13		**Serverová část**	**28 dní**			**72160 Kč**
14		Datová část	3 dní	9	Programátor JS	9840 Kč
15		Aplikační logika	14 dní	14	Programátor JS	45920 Kč
16		Zabezpečení	5 dní	11	Programátor JS	16400 Kč
17		**Klientská část**	**14 dní**			**55760 Kč**
18		Klientská logika	14 dní	15	Programátor JS	45920 Kč
19		Zabezpečení	3 dní	16	Programátor JS	9840 Kč
20		**Testování**	**5 dní**			**21120 Kč**
21		Automatické testy	5 dní	7	Tester	13200 Kč
22		Manuální testování	3 dní	12	Tester	7920 Kč
23		**Dokumentace**	**5 dní**			**21120 Kč**
24		Zdrojového kódu	3 dní	12	IT specialista	7920 Kč
25		Uživatelská	5 dní	12	IT specialista	13200 Kč
26		Nasazení do provozu	1 den	12;23	IT specialista	2640 Kč

Obrázek 6.1: Náklady na vývoj vypočítané v ProjectLibre

Náklady na vývoj systému jsou popsány v tabulce 6.1. Tabulka byla vyexportována jako obrázek z programu ProjectLibre, s jehož pomocí byl proveden odhad nákladů na lidské zdroje. Celkové odhadované náklady na lidské zdroje pro vývoj systému jsou 300 000 Kč. Odhadovanou částu je potřeba navýšit o režijní náklady, které jsou odhadovány na 50 %. Tudíž odhadované náklady na systém budou 450 000 Kč. Časová náročnost projektu je 90 člověkodní. Některé části v projektu lze provádět paralelně, takže při dostatečném množství lidských zdrojů se délka projektu může zkrátit až na 50 dní. Mnoho částí systému je již v této práci vyhotoveno, tudíž náklady na úplné dokončení systému mohou klesnout až na 150 000 Kč.

6.2 Náklady na provoz a údržbu systému

Systém byl navržen tak, aby náklady na jeho údržbu a provoz byly co nejnižší, což je jedna z výhod oproti jiným řešení. Systém bude nutné udržovat pouze v případě, že dojde ke změnám v rozhraní operačního systému RouterOS,v prostředí NodeJS nebo v použitých knihovnách. Takové změny jsou však

vzácný případ a pokud k nim dojde, neměly by být náklady vyšší než jednotky člověkodní, což může být maximálně jednotky tisíc Kč za rok.

Jelikož je systém postaven na otevřených technologiích, umožňujících bezplatné komerční použití, nevznikají s provozem systému žádné jednorázové ani paušální náklady na provoz za předpokladu, že systém bude provozován na serverech, které ISP často již vlastní. Pokud žádný server nevlastní, systém bude možné provozovat na běžném počítači s architekturou a operačním systémem, který podporuje NodeJS a MongoDB, jehož jednorázová pořizovací cena je maximálně v desítkách tisíc Kč.

6.3 Limity vytvořeného řešení

Žádný systém není univerzální a i v tomto systému existují různá omezení. Některá omezení z principu zvolených technologií, jiná z důvodu nutnosti přizpůsobit systém cílovému prostředí.

Řešení tohoto systému je určené pro běh ve webovém prostředí. Z toho vyplývá omezení, že je nutné mít funkční internetové připojení, aby bylo možné schopnosti systému bezproblémově používat. Další omezení je, že není možné přizpůsobit aplikaci tak, aby běžela ve všech prohlížečích z důvodu využití moderních technologií a zabezpečení systému. Aby nebyli uživatelé příliš omezováni, byl pro klientskou část systému zvolen překlad do starších verzí. Pro běh klientské části je nutné mít prohlížeč, který zvládá ECC, HTTP, Websocket, ES5, HTML5, CSS3 a neblokuje cookies.

Dalším omezením je nutnost vytvoření prvního požadavku na požadovanou platbu ručně, od které se budou dále automaticky generovat požadavky na provedení platby v požadované periodě, která bude uvedena ve smlouvě. Pro plnohodnotné využití systému bude nutné vložit záznamy o zařízeních, protože aplikace nebude schopna vytvořit si sama záznamy o zařízeních jen tím, že by se připojila do sítě, kterou by si prozkoumala a přidala si záznamy o existujících zařízeních.

Další vlastností systému je, že veškerá data jsou přístupná všem zaměstnancům. Pokud bude požadováno, aby někteří zaměstnanci neměli přístup ke všem datům, nebudou moci tento systém používat.

6.4 Možnosti budoucího rozvoje systému

Jednou z možností rozšíření funkcionalit systému je automatické prohledání síťových zařízení v síti a automatické přidání do kolekce zařízení včetně jejich rozhraní. Další užitečnou funkcionalitou může být automatický import provedených plateb z CSV souboru, který bude vyexportovaný z bankovního účtu pro příjem plateb za síťové služby.

Systém bude možné rozšířit o další systémové role, například pro zaměstnance, kteří budou mít právo číst záznamy o klientech pouze pokud

s nimi mají uzavřenou smlouvu a k platbám vázaných pouze k jejich smlouvě. Další role v systému může být pro techniky, kteří nebudou mít k datům o klientech přístup z důvodu potřeby utajení citlivých údajů.

Velkým přínosem by byla funkce pro automatickou kontrolu stability sítě a tvorbu grafických reportů, či automatické informování administrátora sítě kdy a kde došlo k výpadku spojení. Další přínos by byl ve vytvoření portálu pro informování klientů o plánovaných údržbách na síti, kvůli kterým může docházet k nestabilitě na síťových spojích.

Systém je navržen tak, aby bylo snadné jej rozšířit o další funkcionality. Po implementaci klientské části bude systém připraven k dalšímu rozvoji, ať už o uvedené části, či o nové požadavky, které mohou vznikat během používání systému.

Závěr

Cílem diplomové práce bylo zanalyzovat a navrhnout CRM systém pro lokálního poskytovatele internetových služeb. Zároveň na základě analýzy a návrhu provést implementaci vybrané části systému. Prvním úkolem bylo provést analýzu požadavků správců sítě na výsledný systém. Na základě provedené analýzy požadavků byla provedena analýza vhodných nástrojů a technologií, ze kterých byly vybrány ty, které jsou vhodné pro vytvoření výsledného systému. Pro dosažení cíle bylo potřeba navrhnout architekturu a design CRM systému. Na základě analýzy a návrhu bylo úkolem provést implementaci prototypu vybrané části systému. Součástí cíle bylo provedení ekonomického vyhodnocení a navržení možností rozšíření systému.

Z provedené analýzy požadavků bylo zjištěno, že malí poskytovatelé internetového připojení mají zájem o systém, který je rozšiřitelný, postavený na volně šiřitelných technologiích a který nevyžaduje zásah do konfigurace již existujících síťových zařízení v síti poskytovatele. Zároveň chtějí, aby systém komunikoval se zařízeními s operačním systémem RouterOS a mohl být nasazen na OS Linux i Windows. Z uživatelského hlediska správci sítě požadují, aby systém měl grafické uživatelské rozhraní pro správce sítě i pro klienty a byl dostupný přes webový prohlížeč, který podporuje EcmaScript5, HTML5, CSS3 a WebSocket. Pro snazší představu, jak se bude systém používat, byly vytvořeny diagramy a k nim přidružené scénáře případů užití. K pochopení vazeb mezi entitami a jejich atributy u poskytovatele byl vytvořen analytický doménový model. Součástí analýzy byl průzkum existujících řešení, při kterém bylo snahou nalézt řešení zcela vyhovující stanoveným funkčním a nefunkčním požadavkům. Při průzkumu nebylo nalezeno žádné řešení, které by bylo z hlediska požadavků zcela vyhovující.

Hlavní přínos této práce spočívá ve vytvoření návrhu informačního systému podle požadavků. V návrhu řešení byla snaha o dosažení flexibilního řešení bez závislosti na zastaralých technologiích a nástrojích. V návrhu řešení tedy byly vybrány a popsány vhodné softwarové nástroje, na kterých bude výsledný systém postaven a které vyhovují daným požadavkům. Vybraným imple-

mentačním jazykem je objektově orientovaný TypeScript, protože přináší výhody v typové kontrole. Kód pro klientskou část systému bude překládán pomocí kompilátoru Babel do standardu EcmaScript5, protože tento standard je mezi webovými prohlížeči široce podporovaný v porovnání s novějšími verzemi. Kód pro serverovou část systému bude překládán do standardu EcmaScript6. Nástroje pro tvorbu uživatelského rozhraní byly vybrány takové, aby bylo snadné pomocí nich tvořit responzivní webdesign, který zajistí správné zobrazení na uživatelských zařízeních s různě velikými obrazovkami.

Další úkol při návrhu řešení spočíval v tvorbě softwarové architektury systému. V diagramu nasazení bylo popsáno, jaký další software navrhovaný systém vyžaduje ke svému běhu. Hlavním uzlem systému je server s prostředím NodeJS. Tato část je rozdělena na serverovou a klientskou aplikaci. Serverová aplikace bude zpracovávat požadavky z klientské aplikace a v případě potřeby bude komunikovat se síťovými zařízeními s RouterOS a s databází MongoDB. Komunikace mezi klientskou a serverovou aplikací bude probíhat po šifrovaném kanálu s použitím dostatečně odolných algoritmů. Pro detailní pohled do softwarové architektury systému byl vytvořen diagram komponent. Ten znázorňuje komunikační směry komponent a zobrazuje informaci, na jakých knihovnách bude hlavní logika každé komponenty postavena.

Ke snadnému pochopení ukládaných dat do databáze byl vytvořen objektový datový model. Na něm jsou znázorněny vazby, podle kterých jsou dále tvořeny dotazy předávané databázi, pokud zasahují do více datových kolekcí. Jeho součástí je definice datového typu u každého atributu. Tento model je využit i v serverové aplikaci systému pro práci s daty.

Protože serverová logika je podle architektury od klientské logiky oddělena, tak jako součást návrhu byla i definice aplikačního programového rozhraní serverové aplikace. To je založeno převážně na HTTP a jeho metodách. V případě, že rozhraní vyžaduje obousměrnou komunikaci, je využit protokol WebSocket. Oba protokoly byly navrženy pro využití v šifrovaném kanálu. Tyto protokoly byly vybrány z důvodu jejich běžné podpory ze strany webových prohlížečů.

V závěrečné části návrhu řešení byly vytvořeny modely GUI v aplikaci Adobe XD. Ty byly rozděleny na klientské a administrátorské rozhraní, některé části jsou však společné pro oba typy uživatelů systému. Podle tohoto návrhu bude vytvořeno GUI výsledné klientské části systému.

Hlavní náplní implementace byla tvorba serverové části systému. Před jejím zahájením bylo nutné vytvořit konfiguraci pro NodeJS. Ta zahrnuje seznam potřebných balíčků pro běh aplikace a skripty pro operace s kódy a výslednými produkty implementace. Důležitou částí je konfigurace pro Webpack. V té bylo nastaveno, jakým způsobem má probíhat překlad a co bude výsledným produktem po překladu. Konfigurace byla vytvořena tak, že není vázaná pouze na tento projekt, ale lze ji využít i pro nové projekty a lehce ji přizpůsobit pro využití dalších nástrojů, které se mohou objevit během dalších let.

Při samotné implementaci byly vytvořeny funkce pro obsluhu HTTPS a WSS požadavků s použitím knihovny Express.js a Socket.io. Ke komunikaci

s databází byly vytvořeny funkce pro operaci s jednotlivými dokumenty nad svými datovými kolekcemi. Dále byly vytvořeny funkce pro autentizaci a autorizaci uživatele, byznys logiku a komunikaci se síťovými zařízeními. Každá funkce byla zařazena do některé komponenty v systému, aby bylo možné kód snadno upravovat podle požadavků a byla dodržena nízká provázanost výsledného kódu. Implementace byla dovedena do stavu, ve kterém je připravena k implementaci klientské části systému. V aktuální podobě má přibližně 3 000 řádků kódu ve 28 logicky oddělených souborech se zdrojovým kódem, které se nacházejí na přiloženém médiu spolu s konfiguračními soubory pro celý projekt. Z implementace klientské části mohou ještě vyplynout důvody k drobným úpravám i v serverové části.

V části diplomové práce, která se zabývá vyhodnocením projektu, byl proveden odhad finančních nákladů na celý projekt. Z těchto odhadovaných nákladů je možné značnou část odečíst, protože mnoho částí již v této práci bylo provedeno. Ve výsledku byly náklady na dokončení a zprovoznění celého systému vyčísleny na 150 000 Kč. Náklady na provoz a údržbu jsou minimální a s většinou si poradí zkušený administrátor u samotného poskytovatele, což je další výhoda navrženého systému. Pokud se objeví nové požadavky na systém, nebude vyžadováno přeprogramování již vytvořených částí díky rozšiřitelné architektuře systému. Součástí vyhodnocení projektu je popis omezení vybraného řešení, které vyplývá z toho, že bylo zvoleno řešení ve formě webové aplikace, která vyžaduje funkční internetové připojení. V této části bylo myšleno i na možnosti budoucího rozšíření systému. Veliký přínos by přineslo rozšíření systému o funkce pro automatickou kontrolu stability komunikačních spojů a informování správců sítě o vyskytujících se anomáliích. Takové rozšíření bude možné do systému snadno implementovat díky rozšiřitelné architektuře.

Jelikož byla provedena implementace jen části systému, bude dále pokračováno v dokončení celého systému včetně jeho řádného otestování. Po dokončení první verze celého systému je plánováno jeho předání správcům sítě, kteří jej nasadí do provozu v nejbližší možné době. Do budoucna je plánováno vytvořit na základě tohoto systému komplexní informační systém pro společnosti, které poskytují internetové služby a nemají automatizované nástroje pro řešení opakujících se operací v síti.

Tento systém může v konečném důsledku pomoci se zkvalitněním poskytovaných služeb tím, že pomůže s detekcí nestabilních spojů v síti. Nestabilita síťových spojení je také častým důvodem odchodu klientů ke konkurenčním poskytovatelům internetových služeb, tudíž tento systém může přispět k udržení stávajících klientů a zvýšení spokojenosti stávajících klientů s poskytovanými službami. Dále systém může v budoucnu přispět k šetření času správců při rutinních operacích na síti díky rozšiřitelnosti systému tak, že si mohou sami přidat do systému další nástroje pro konfiguraci zařízení. Současně díky ucelenému přehledu všech zařízení v síti poskytovatele internetových služeb bude snadné tvořit plány pro postupnou obnovu zastaralých síťových zařízení za novější, která zajišťují lepší stabilitu spojení. Systém díky evidenci plateb

usnadní dohledání chybějících plateb za poskytované služby.

Společně se zhotovením této práce si její autor rozšířil znalosti v oblasti tvorby moderních webových aplikací. Rozšířil si znalosti o nástrojích pro tvorbu uživatelského rozhraní webových aplikací. Dále si rozšířil znalosti o komunikační protokoly, nástroje, knihovny a jazyky pro vývoj aplikací. Touto prací si autor vyzkoušel práci softwarového analytika, designéra, architekta i programátora, čímž si vzájemně propojil dílčí znalosti získané během studia na vysoké škole.

Literatura

[1] ARLOW, J.; NEUSTADT, I.: *UML 2 a unifikovaný proces vývoje aplikací*. Brno: Computer Press, druhé vydání, 2011, ISBN 978-80-251-1503-9.

[2] Žoltá, L.: UML (Unified Modeling Language) [online]. [cit. 2019-08-24]. Dostupné z: http://lucie.zolta.cz/index.php/iformacni-systemy-databaze/33-uml-unified-modeling-language

[3] Dutchguilder: Iterative development illustration [online]. [cit. 2019-08-23]. Dostupné z: https://commons.wikimedia.org/wiki/File:Development-iterative.png

[4] Žoltá, L.: Disciplína sběr a analýza požadavků [online]. [cit. 2019-08-05]. Dostupné z: http://lucie.zolta.cz/index.php/softwarove-inzenyrstvi/150-disciplina-sber-a-analyza-pozadavku

[5] SOMMERVILLE, I.: *Softwarové inženýrství*. Brno: Computer Press, první vydání, 2013, ISBN 978-80-251-3826-7.

[6] SHARKIE, C.; FISHER, A.: *Responzivní webdesign: okamžitě*. Brno: Computer Press, první vydání, 2015, ISBN 978-80-251-4384-1.

[7] KROENKE, D. M.; AUER, D. J.: *Databáze*. Brno: Computer Press, první vydání, 2015, ISBN 978-80-251-4352-0.

[8] Ing. Michal VALENTA, Ph.D.: DBS – Databázové modely [online]. [cit. 2019-09-20]. Dostupné z: https://users.fit.cvut.cz/valenta/doku/lib/exe/fetch.php/bivs/dbs_02_databazove_modely.pdf

[9] DANĚK, J.: Protokol HTTP a WWW servery [online]. [cit. 2019-10-22]. Dostupné z: https://www.fi.muni.cz/~kas/pv090/referaty/2013-jaro/web.html

[10] Kaazing ©: The WebSocket Protocol [online]. [cit. 2019-10-22]. Dostupné z: http://www.websocket.org/aboutwebsocket.html

[11] KUBA, M.: Tutoriál Web Services [online]. [cit. 2019-10-22]. Dostupné z: https://dior.ics.muni.cz/~makub/soap/tutorial.html

[12] RÁBOVÁ, I.: Globální architektura IS/IT [online]. [cit. 2019-09-21]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=5115

[13] BRUCKNER, T.; a spol.: *Tvorba informačních systémů: principy, metodiky, architektury*. Praha: Grada, první vydání, 2012, ISBN 978-80-247-4153-6.

[14] HANÁČEK, P.: Bezpečnostní funkce v počítačových sítích. *Zpravodaj ÚVT MU. [online]*, , č. 2, 1999: s. 5–9, ISSN 1212-0901, [cit. 2019-09-23]. Dostupné z: http://webserver.ics.muni.cz/bulletin/articles/171.html

[15] FOLTÝNEK, T.; PŘICHYSTAL, J.: Komprimace a šifrování [online]. [cit. 2019-09-22]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/index.pl?opora=621

[16] ČERMÁK, M.: Autentizace: Jak v systému bezpečně uložit heslo [online]. [cit. 2019-09-25]. Dostupné z: https://www.cleverandsmart.cz/autentizace-jak-v-systemu-bezpecne-ulozit-heslo/

[17] ZONER software a.s.: Podpora ECC certifikátů [online]. [cit. 2019-09-23]. Dostupné z: https://blog.sslmarket.cz/inpage/podpora-ecc-certfiikatu/

[18] JANSA, L.; OTEVŘEL, P.; ŠTEVKO, M.: *Softwarové právo*. Brno: Computer Press, třetí vydání, 2018, ISBN 978-80-251-4914-0.

[19] The Eclipse Foundation: Eclipse Papyrus ™ [online]. [cit. 2019-08-19]. Dostupné z: https://www.eclipse.org/papyrus/

[20] Oracle Corporation and/or its affiliates: OpenJDK build [online]. [cit. 2019-08-19]. Dostupné z: https://github.com/ojdkbuild/ojdkbuild

[21] ExpertBilling: Fakturační systém ExpertBilling [online]. [cit. 2019-10-07]. Dostupné z: http://expertbilling.ru/

[22] Splynx s.r.o.: Splynx Framework [online]. [cit. 2019-10-07]. Dostupné z: https://splynx.com/

[23] MUPSSOFT: MUPSSOFT.COM [online]. [cit. 2019-10-07]. Dostupné z: https://www.mupssoft.com/

[24] ICONWAVE TECHNOLOGIES PVT. LTD.: Transformative Technologies for your Telecommunications, ISP & Media Business [online]. [cit. 2019-10-07]. Dostupné z: https://www.iconwavetech.com/hotspot-billing-software

[25] ISP Cube: Simple ISP Administration [online]. [cit. 2019-10-07]. Dostupné z: https://www.ispcube.com/en/

[26] Magnaquest Technologies Ltd.: SURE! Broadband Billing & CRM Solution [online]. [cit. 2019-10-11]. Dostupné z: https://www.magnaquest.com/

[27] Anatod ®: NETWORK MANAGEMENT [online]. [cit. 2019-10-11]. Dostupné z: https://www.anatod.com/en/

[28] JBRSOFT: TUCANA ISPERP [online]. [cit. 2019-10-11]. Dostupné z: http://isperp.org/features.html

[29] Antamedia: ISP BILLING [online]. [cit. 2019-10-11]. Dostupné z: https://www.antamedia.com/isp-billing/

[30] JAZE NETWORKS PVT LTD.: Jaze ISP Manager [online]. [cit. 2019-10-11]. Dostupné z: https://www.jazenetworks.com/jaze-isp-manager/

[31] KVAPIL, J.: Úvod do TypeScriptu [online]. [cit. 2019-10-17]. Dostupné z: https://www.itnetwork.cz/javascript/typescript/uvod-do-typescriptu

[32] MÁCA, J.: Úvod do Node.js [online]. [cit. 2019-10-17]. Dostupné z: https://www.itnetwork.cz/javascript/nodejs/uvod-do-nodejs

[33] SEDLÁČEK, P.: Úvod do MongoDB [online]. [cit. 2019-10-17]. Dostupné z: https://www.itnetwork.cz/javascript/nodejs/uvod-do-mongodb

[34] Microsoft: Vytvoření aplikace v Node. js a Express v aplikaci Visual Studio [online]. [cit. 2019-10-17]. Dostupné z: https://docs.microsoft.com/cs-cz/visualstudio/javascript/tutorial-nodejs?view=vs-2017

[35] KURIAN, A. M.: How to build a real time chat application in Node.js using Express, Mongoose and Socket.io [online]. [cit. 2019-10-17]. Dostupné z: https://www.freecodecamp.org/news/simple-chat-application-in-node-js-using-express-mongoose-and-socket-io-ee62d94f5804/

[36] HANSON, J.: Passport [online]. [cit. 2019-10-17]. Dostupné z: http://www.passportjs.org/docs/

[37] ERDELJAC, A.: Learn how to handle authentication with Node using Passport.js [online]. [cit. 2019-10-17]. Dostupné z: https://www.freecodecamp.org/news/learn-how-to-handle-authentication-with-node-using-passport-js-4a56ed18e81e/

[38] BITTNER, J.: Úvod do CSS preprocesoru Sass [online]. [cit. 2019-10-17]. Dostupné z: https://www.itnetwork.cz/html-css/webove-portfolio/tutorial-moderni-webove-portfolio-sass

[39] Plotly: Plotly JavaScript Open Source Graphing Library [online]. [cit. 2019-10-17]. Dostupné z: https://plot.ly/javascript/

[40] POTTER, J.: React vs Vue vs @angular/core [online]. [cit. 2019-10-17]. Dostupné z: https://www.npmtrends.com/react-vs-vue-vs-@angular/core

[41] MIKŠŮ, V.: React – Úvod [online]. [cit. 2019-10-17]. Dostupné z: https://www.dzejes.cz/react-uvod.html

[42] MIKŠŮ, V.: JavaScript? Babel. [online]. [cit. 2019-10-17]. Dostupné z: https://www.dzejes.cz/babel.html

[43] MIKŠŮ, V.: Nástroje [online]. [cit. 2019-10-17]. Dostupné z: https://www.dzejes.cz/nastroje.html

[44] Webpack.js: Webpack module bundler [online]. [cit. 2019-10-17]. Dostupné z: http://webpack.github.io/

[45] MIKŠŮ, V.: První dev stack [online]. [cit. 2019-10-17]. Dostupné z: https://www.dzejes.cz/prvni-dev-stack.html

[46] Microsoft: Visual Studio Community [online]. [cit. 2019-11-11]. Dostupné z: https://visualstudio.microsoft.com/cs/vs/community/

Příloha **A**

Seznam použitých zkratek

AES Advanced Encryption Standard

AIO All In One

AJAX Asynchronous JavaScript and XML

API Application Program Interface

BSD Berkeley Software Distribution

CPAL Common Public Attribution License

CRM Customer Relationship Management

CRUD Create Read Update Delete

CSS Cascading Style Sheets

CSV Comma-separated Values

DB Database

DBMS Database Management System

DBS Database System

DCL Data Control Language

DDL Data Definition Language

DML Data Manipulation Language

DOM Document Object Model

DQL Data Query Language

ECC Elliptic Curve Cryptography

ERP Enterprise Resource Planning

ES ECMAScript

FOSS Free and Open Source Software

GDPR General Data Protection Regulation

GIS Geografic Information System

GPL General Public License

GUI Graphical User Interface

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secured

IDE Integrated Development Environment

IP Internet Protocol

IS Informační Systém

ISP Internet Service Provider

JS JavaScript

JSON JavaScript Object Notation

JSX JavaScript XML

KPI Key Performance Indicator

LTS Long Term Support

MIT Massachusetts Institute of Technology

MRP Material Resource Planning

MS Microsoft

OID Object Identifier

OQL Object Query Language

ORM Object Relational Mapper

OSS Open Source Software

REST Representational State Transfer

RIS Reservation Information System

RSA Rivest, Shamir, Adleman

SCM Supply Chain Management

SHA Secure Hash Algorithm

SOAP Simple Object Access Protocol

SPA Single Page Application

SQL Structured Query Language

SSH Secure Shell

SSL Secure Sockets Layer

SVG Scalable Vector Graphics

TCL Transaction Control Language

TLS Transport Layer Security

TS TypeScript

UI User Interface

UML Unified Modeling Language

UP Unified Process

URI Uniform Resource Identifier

XML Extensible Markup Language

WS WebSocket

WSS WebSocket Secured

Příloha **B**

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `config` ..... konfigurace testovacích síťových zařízení
- `obr` ..... obrázky
  - `UI` ..... obrázky GUI
- `src`
  - `impl` ..... zdrojové kódy systému
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_Korel_Lukas_2020.pdf` ..... text práce ve formátu PDF
  - `DP_Korel_Lukas_2020_zadani.pdf` ..... zadání práce ve formátu PDF