ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Veremchuk** Jméno: **Mykhailo** Osobní číslo: **474371**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Systém pro podporu Patch a Asset managementu

Název bakalářské práce anglicky:

Patch and Asset Management System

Pokyny pro vypracování:

Vytvořte aplikaci pro podporu Patch a Asset managementu. Postupujte následovně:
1) Analyzujte problematiku Patch a Asset managementu.
2) Zmapujte existující podpůrné systémy a vyhodnoťte jejich použitelnost.
3) Identifikujte reálné potřeby vybraného zadavatele (dodá vedoucí práce) a namapujte je na systémy, hodnocené v předchozím bodě. Zdůvodněte smysluplnost implementace nového systému.
4) Navrhněte a implementujte nový systém, který bude podporovat:
- pro Asset management:
• evidenci zařízení zákazníků, pro která se poskytuje služba patch managementu;
• možnost řazení zařízení do skupin.
- pro Patch management:
• evidenci produktů (SW/HW) a jejich přiřazení konkrétnímu zařízení;
• evidenci, včetně historie, produktových patchů;
• nastavení produktových časovačů, po jejichž uplynutí se v systému Jira vytvoří odpovědné osobě tickety, upozorňující na nutnost kontroly nových patchů.
5) Systém ověřte formou uživatelského testování pro scénáře, které definujete se zadavatelem ve fázi návrhu.

Seznam doporučené literatury:

[1] IT Asset Management | Atlassian. Atlassian | Software Development and Collaboration Tools [online]. Copyright © 2019 Atlassian. Dostupné z: https://www.atlassian.com/itsm/it-asset-management
[2] Patch Management - an overview | ScienceDirect Topics. ScienceDirect.com | Science, health and medical journals, full text articles and books. [online]. Copyright © 2019 Elsevier B.V. or its licensors or contributors.Dostupné z: https://www.sciencedirect.com/topics/computer-science/patch-management

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Náplava, Ph.D., katedra ekonomiky, manažerství a humanitních věd FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **14.02.2020** Termín odevzdání bakalářské práce: **22.05.2020**

Platnost zadání bakalářské práce: **30.09.2021**

Ing. Pavel Náplava, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.	
Datum převzetí zadání	Podpis studenta

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Systém pro podporu Patch a Asset managementu

Bakalářská práce

Mykhailo Veremchuk

Studijní program: Softwarové inženýrství a technologie
Vedoucí: Ing. Pavel Náplava, Ph.D.

Praze, 2020

Vedoucí:
Ing. Pavel Náplava, Ph.D.
Katedra ekonomiky, manažerství a humanitních věd
Fakulta elektrotechnická
České vysoké učení technické v Praze
Technická 2
160 00 Praha 6
Česká republika

Prohlášení

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

V Praze, 2020

.....
Mykhailo Veremchuk

Poděkování

Rád bych poděkoval vedoucímu mé práce panu Ing. Pavlu Náplavovi, Ph.D. za jeho věcné připomínky a vstřícný přístup. Také děkuji své rodině za jejich podporu po celou dobu mého studia.

Abstrakt

Cílem této práce bylo analyzovat obecnou problematiku Patch a Asset managementu, identifikovat reálné potřeby a procesy externího zadavatele, prozkoumat existující podpůrné systémy a vyhodnotit jejich praktickou použitelnost. Následně podle detailně propracovaných požadavků byl navržen a implementován systém. Funkčnost aplikace se ověřila formou uživatelských testů.

Klíčová slova: Webová aplikace, IT asset management, aktualizace softwaru, analýza, patch, asset.

Abstract

The aim of this work was to analyze the general issues of patch and asset management, identify the real needs and processes of the external client, examine the existing systems, and evaluate their practical applicability. Subsequently, to design and implement a system that will satisfy the client. The functionality of the application was verified in the form of user tests.

Keywords: Web application, IT asset management, patch, Patch management software, analysis, asset.

Title translation: Patch and Asset Management System

Seznam tabulek

Seznam obrázků

Seznam zkratek

AM Asset Management. 3

ICT Information and communications technology. 7

ITAM Information Technology Asset Management. 4

PM Patch Management. 4

SPA Single Page Application. 29

SW Software. 3

TSA Technical security architect. 8

UML Unified Modeling Language. 26

Obsah

Úvod

Počet bezpečnostních hrozeb a zranitelností v dnešní době neustále roste. Drtivá většina výrobců softwaru nabízí bezpečnostní aktualizace, které opravují případné chyby a snižují celkovou zranitelnost produktu. Je na nás, abychom si pravidelně zajišťovali aktualizaci těchto programů, čímž značně zmenšíme bezpečnostní rizika a zranitelnost celého prostředí. Pro společnosti, které poskytují účinnou kybernetickou obranu, je čím dále tím těžší mít přehled o aktuálním stavu softwaru nainstalovaného u zákazníka. V této práci se zabývám Asset a Patch Managementem, které řeší výše zmíněné problémy. Hlavním cílem této práce je provést analýzu požadavků externího zadavatele a následně navrhnout a implementovat systém.

Toto téma jsem si zvolil z několika důvodů. Za prvé zkusit navrhnout a implementovat škálovatelný systém s využitím nových technologií. Za druhé protože se již delší dobu zabývám patch managementem. Nakonec realizovat projekt pro externího zadavalete, pro kterého pracuji na pozici junior programátor.

Práci jsem rozdělil do osmi kapitol. První kapitola definuje klíčové pojmy této práce a popisuje obecnou problematiku Patch a Asset managementu. Další kapitola představuje externího zadavatele projektu a určuje základní požadavky na systém. Třetí kapitola se zabývá analýzou a porovnáváním již existujících aplikací na trhu a vyhodnocením jejich použitelností. Čtvrtá kapitola detailněji zkoumá procesy zadavatele a specifikuje požadavky na systém. Další kapitola se zabývá návrhem aplikace, včetně popisu architektury, použitých technologií a UML diagramy. Šestá kapitola popisuje průběh implementace a představuje samotnou aplikaci. Další kapitola se zabývá testováním systému. Poslední kapitoly práce vyhodnocují projekt a popisují jeho aktuální stav.

Kapitola 1

Definice pojmů

V této kapitole uvádím klíčové pojmy spojené s tématem mé bakalářské práce. Také detailněji analyzuji problematiku Patch a Asset managementu a jejich využití v praxi. Kvůli neexistenci vhodného českého překladu pojmů definovaných v této části práce, po dohodě s vedoucím práce jsem se rozhodl používat jejich anglické názvy.

Asset

Assety jsou zdroje s ekonomickou hodnotou, které jednotlivec, korporace nebo libovolná organizace vlastní nebo kontroluje s očekáváním, že poskytnou budoucí užitek. Assetem může být pohledávka, cenný papír, zásoba, goodwill, vybavení, Software (SW), hardware, stroj nebo nemovitost[1].

V tomto dokumentu se pod pojmem asset rozumí konkrétní fyzické nebo virtuální zařízení, pro které se realizuje služba patch managementu.

Asset Management (AM)

Asset management je plánování, řízení a údržba dlouhodobých assetů, investic, za která je skupina nebo entita odpovědná během celého jejich životního cyklu. Může se vztahovat jak na hmotné assety (fyzické předměty, jako jsou budovy nebo zařízení), tak na nehmotná assety (jako je duševní vlastnictví, lidský kapitál apod.).

Asset management je systematický proces vývoje, provozu, údržby, modernizace a vyřazování assetů co nejefektivnějším způsobem (včetně všech nákladů, rizik a atributů výkonu)[2].

Typy asset managementu

Z definice asset managementu vyplývá, že se jedná o širokou kategorii řízení, která zahrnuje několik různých oborů a odvětví[3]:

- **Správa finančních assetů**

  Finanční správa assetů je sektorem finančních služeb, který spravuje investiční fondy a klientské investiční účty.

- **Správa podnikových assetů**

  Správa dlouhodobého majetku organizace včetně pořízení, provozu, údržby a vyřazení z provozu. Definice se rozšiřuje i o nehmotné assety.

- **Správa veřejné infrastruktury**

  Správa dopravní infrastruktury (pozemní komunikace, mosty, letiště), občanského vybavení (stavby a pozemky sloužící k prospěchu společnosti, jako jsou vzdělávací zařízení, správní a administrativní objekty) se zaměřením na údržbu, obnovu a výměnu infrastruktury.

- **Správa dlouhodobých assetů**

  Sledování dlouhodobého majetku pro účely finančního účetnictví, údržby a prevence ztrát.

- **Správa digitálních assetů**

  Správa informací, které organizace vlastní, řídí nebo má právo používat.

- **Správa IT assetů**

  Řízení, kontrola hardwaru a softwaru. Správa IT assetů zahrnuje funkce údržby, správy smluv a účetnictví pro IT assety.

IT asset management

V této práci se budu hlavně věnovat problematice IT asset managementu.

Podle Mezinárodní asociace správců assetů pro IT (IAITAM) je Information Technology Asset Management (ITAM) soubor obchodních praktik, které zahrnují assety IT ve všech obchodních jednotkách organizace. Spojuje se s finančními, inventárními a smluvními povinnostmi správy za řízení celkového životního cyklu těchto assetů, včetně taktického a strategického rozhodování[4].

Investice do IT Asset Managementu poskytuje podstatné a měřitelné výhody pro krátkodobé a dlouhodobé potřeby a cíle.

Patch

Patch je sada změn softwarového produktu nebo podpůrných dat určených k jeho aktualizaci, opravě nebo zlepšení. Tyto změny zahrnují opravu bezpečnostních chyb, novou funkcionalitu a vylepšení funkčnosti, použitelnosti a nebo výkonu aplikace[5].

Kvůli zjednodušení se pod pojmem patch v tomto dokumentu rozumí nejen oprava, ale i nová verze aktuální produktové větve. Jde tedy o všechny vydané aktualizace sledovaného produktu na úrovni patch managementu.

Patch Management (PM)

Patch management je strategie pro správu oprav a aktualizací softwarových aplikací a technologií. Patche softwaru jsou často nutné k vyřešení stávajících problémů se softwarem, které byly nalezeny v předchozích verzích. Mnoho z těchto oprav souvisí s bezpečností a zranitelností produktu. Jiné mohou souviset s rozšířením funkčností softwaru[6].

Patch je nedílnou součástí zajištění výkonu a stability operačních systémů a aplikací. Aktualizace softwaru řeší problémy s výkonem a odstraňují zastaralé funkce. Zavedení patch managementu pomáhá podnikům nebo organizacím bezpečně uchovávat zákaznická a jiná data a zabránit narušení bezpečnosti systémů a celé společnosti[7].

Shrnutí kapitoly

V této kapitole jsem představil problematiku Asset a Patch Managementu. V následujících kapitolách budou popsány požadavky zadavatele na systém.

Kapitola 2

Představení externího zadavatele

Zadavatelem mé bakalářské práce je kyberbezpečnostní společnost Corpus Solutions a.s., která se specializuje na problematiku Information and communications technology (ICT) Security. Společnost úspěšně působí na trhu informačních a komunikačních technologií již od roku 1992[8].

„*Je stabilním partnerem pro dlouhodobou spolupráci založenou na porozumění potřebám zákazníků, pomáhá, aby jejich business aplikace pracovaly v infrastrukturním prostředí tak, aby byla maximálním způsobem garantována jejich bezpečnost, výkonnost a efektivita provozu. Projekty jsou realizovány spolu s konzultačními službami, které je pomáhají zasadit do celkového systému procesů a řízení IT služeb v prostředí zákazníka. Tím je zajištěna cílovost projektů a sladění s obchodními a podnikatelskými záměry.*“[9]

S nárůstem služeb poskytovaných společností se zvyšuje počet podpurných interních systémů. V praxi to pak znamená, že potřebné informace jsou uloženy na různých místech a navigace v těchto systémech je stále obtížnější.

Jako primární systém pro řízení projektů a evidenci úkolů/chyb zaměstnanci využívají Jiru od společnosti Atlassian, Inc. „*JIRA podporuje a usnadňuje proces řízení projektů a požadavků, nabízí flexibilní a uživatelské nástroje pro řízení a sledování pracovníků při výkonu plnění úkolů. JIRA je orientován na podporu dosažení očekávaného výkonu na projektu.*“[10]

Další nástroj, který pracovníci z Corpusu používají, je Confluence také od společnosti Atlassian, Inc. Confluence je nástroj pro spolupráci v oblasti dokumentace, který pomáhá týmům spolupracovat a sdílet znalosti efektivně. V Confluence je obsah vytvářen a organizován pomocí prostoru, stránek a blogů.

Servisní tým společnosti již několik měsíců má v plánu vylepšit a optimalizovat stávající interní systém pro podporu Asset a Patch Managementu. Důvodem bylo jednak omezená funkcionalita systému, jednak velmi nízká uživatelská přívětivost a slabá spolehlivost. Proto členy servisního týmu misto aplikaci raději používají tabulky uvnitř prostorů Confluence.

2.1 Základní požadavky na systém

Všechny schůzky se konaly s ředitelem servisního týmu. Po několika setkáních vznikla základní specifikace, kde byly stanovené cíle patch a asset managementu a základní požadavky na systém.

Cílem AM je přehled, správa a evidence konkrétních fyzických nebo virtuálních zařízení, pro která se realizuje služba patch managementu.

Hlavní cíl PM je udržovat produkční systémy aktualizované primárně vzhledem k bezpečnostním hrozbám, sekundárně s ohledem na stabilitu, v poslední řadě kvůli dostupnosti nových funkcí. Patch management je realizován na úrovni jednotlivých vývojových větví spravovaných produktů, tj. do procesu PM patří nasazování kritických oprav, drobných fixů, větších oprav (service pack, hotfix akumulator) a ve spolupráci s projektovým týmem a Technical security architect (TSA) i podvětví spravovaných produktů. Pod proces PM nepatří nasazování nových vývojových větví produktů.

Základní požadavky na systém jsem rozdělil do dvou kategorií: obecné požadavky a klíčové systémové požadavky.

Obecné požadavky na systém:

- Jedním z hlavních požadavků je provázanost AM a PM. To znamená, aby systém umožňoval pro jakýkoli asset zobrazit aktuální verzi softwaru (patch), který je na tomto assetu nainstalován, případně zobrazit, jestli existuje novější.
- Dalším požadavkem je možnost propojení s jinými systémy. Pro splnění tohoto požadavku musí systém poskytovat otevřené API.

Klíčové systémové požadavky:

- Základním systémovým požadavkem je možnost zařazení assetů do skupin. Servisní tým společnosti Corpus Solutions a.s. má velké množství zákazníků, kterým poskytuje účinnou kybernetickou obranu, jejíž důležitou částí je instalace a údržba assetů (fyzických nebo virtuálních zařízení). Proto je pro bezpečnostní experty z Corpusu nezbytné vědět, které assety jsou nainstalované u zákazníků, a mít přehled o jejich aktuálním stavu.
- Dalším systémovým požadavkem je správa vlastních patchů. To znamená, že systém umožní přidávat, aktualizovat a mazat patche, které si pak uživatel bude moci spojit s assetem.

Na dalších schůzkách se zadavatelem jsem se zaměřil na specifikaci systémových požadavků a rozšíření aplikací o novou funkčnost (kapitola 4).

Kapitola 3

Analýza dostupných systémů

Na základě požadavků zadavatele na systém (podkapitola 2.1) byla v rámci této kapitoly provedena analýza dostupných řešení patch a asset managementu. Analýzu jsem provedl hlavně proto, abych nevyvíjel něco, co již na trhu existuje, a také pro inspiraci dostupnými řešeními při návrhu vlastního systému.

Pro vyhledávání pojmů jsem použil Google. Veškeré texty pro vyhledávání byly do vyhledávače zadané v anglickém jazyce kvůli větsímu počtu relevantních výsledků.

3.1 Výsledky vyhledávání pojmu IT Asset and Patch management

Jelikož jedním ze základních požadavků na systém je propojenost patch a asset managementu (podkapitola 2.1), pro vyhledávání jsem použil pojem „*IT Asset and Patch management.*“

Bohužel vyhledání výše zmíněného pojmu nenašlo žadnou aplikaci, která by splňovala základní požadavky.

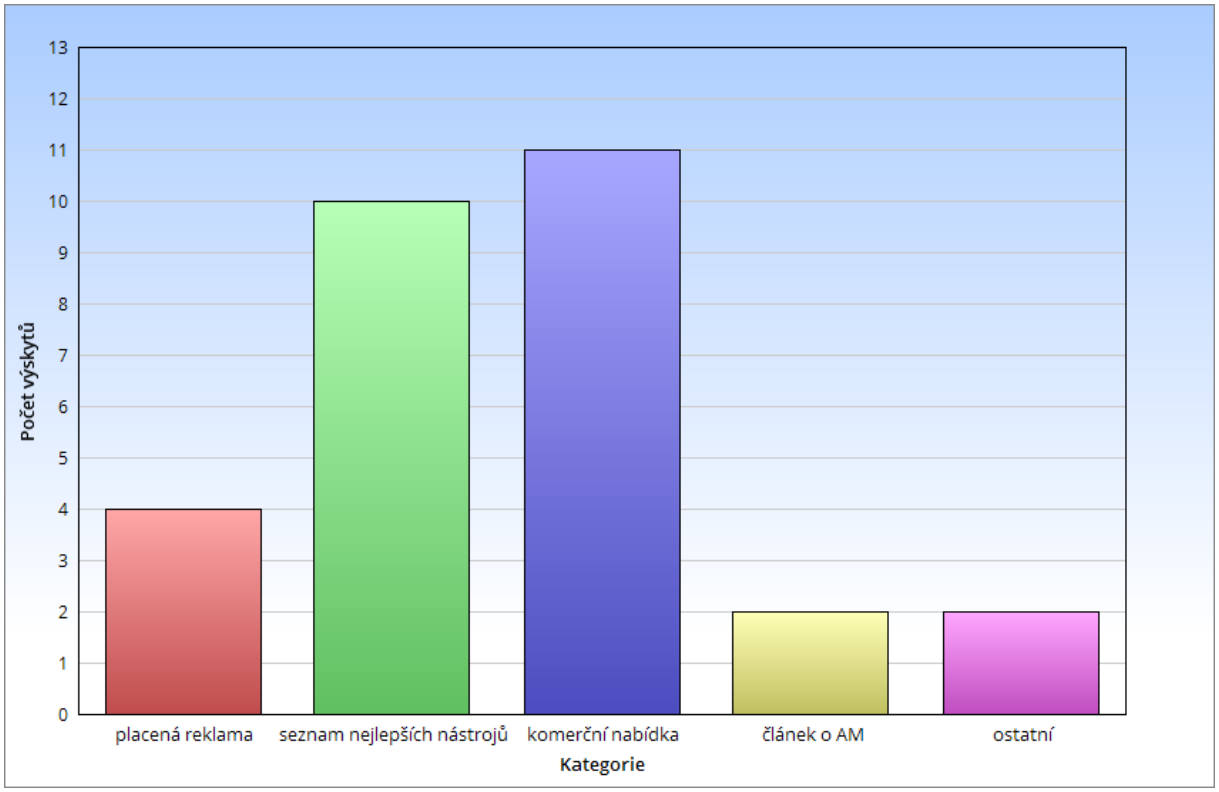
3.2 Výsledky vyhledávání pojmu IT Asset management software

Vyhledávání pojmu „*IT Asset management*“ našlo malý počet relevantních odkázů, proto jsem se rozhodl tento pojem rozšířit o slovo „*software*“.

Do analýzy jsem započítal prvních 30 nalezených (cca 5 stránek Google) výsledků, které jsem rozdělil do následujících kategorií:

- placená reklama - reklamy softwarů pro podporu asset managementu poskytované společností Google
- seznam nejlepších nástrojů - články, které porovnávají komerční nabídky ITAM softwarů
- komerční nabídka - webové stránky výrobců asset managementu
- článek o ITAM - internetové články o obecné problematice asset managementu
- ostatní - články, které nesouvisí s vyhledávacím pojmem

Na obrázku 3.1 je uveden počet výskytů nalezených záznamů podle jednotlivých kategorií. Pro lepší pochopení jsou výsledná data reprezentována formou sloupového grafu.



Obrázek 3.1: Analýza vyhledávání pojmu IT Asset management software.

Po analýze komerčních řešení, jsem zvolil pro detailnější zkoumání a porovnávání systémy, které splňují alespoň část základních požadavků. Abych získal potřebné informace o produktu, vždy jsem používal webové stránky výrobce.

Následující tabulka 3.2 ukazuje společné a odlišné funkčnosti vybraných systémů a jestli splňují základní požadavky zadavatele na systém (znázorněné tučným písmem). Ostatní vlastnosti systému jsem spolu s ředitelem servisního týmu vybíral podle jejich možného budoucího využití ve společnosti Corpus Solutions a.s.

Název produktu/ Vlastnost systému	Skenování assetů	Vzdálený přístup k assetům	Strom assetů	**Otevřené API**	**Podpora Patch managementu**	**Seskupení assetů**	Cena
Asset Explorer [11]	ANO	ANO	NE	ANO	NE	ANO	2995$
Insight Asset management[12]	NE	NE	ANO	ANO	NE	NE	1000$
Fresh service [13]	ANO	ANO	NE	ANO	NE	ANO	2505$
Asset panda [14]	NE	NE	NE	ANO	NE	NE	1800$

Tabulka 3.1: Porovnání komerčních nabídek pro Asset Management

Vysvětlení jednotlivých sloupců tabulky:

- Skenování assetů - systém je schopen oskenovat a rozpoznat různé druhy assetů v rámci sítě.
- Vzdálený přístup k assetům - systém dokáže se vzdáleně připojit na asset (virtuální/fyzické zařízení), případně ho restartovat.
- Strom assetů - systém je schopen zobrazit assety ve formě grafického stromu.
- Otevřené API - systém má otevřené API pro integraci s jinými systémy.
- Podpora Patch managementu - aplikace umožňuje provázat asset se softwarovým produktem.
- Seskupení assetů - systém poskytuje zařazení assetů do jednotlivých skupin.
- Cena - cena je spočítaná pro 1000 assetů na rok. Aktuálně servisní tým Corpus Solutions a.s. má na starosti kolem 750 assetů. V případě Insight Asset managementu uvedená částka je pro 20 zaměstnanců (velikost servisního týmu) a neomezený počet assetů na rok. Vyhledávání cen bylo provedené 21.12.2019.

Jak je vidět z tabulky, žádná ze zkoumaných mnou aplikací nepodporuje patch management, což je jedním z hlavních požadavků na systém (podkapitola 2.1).

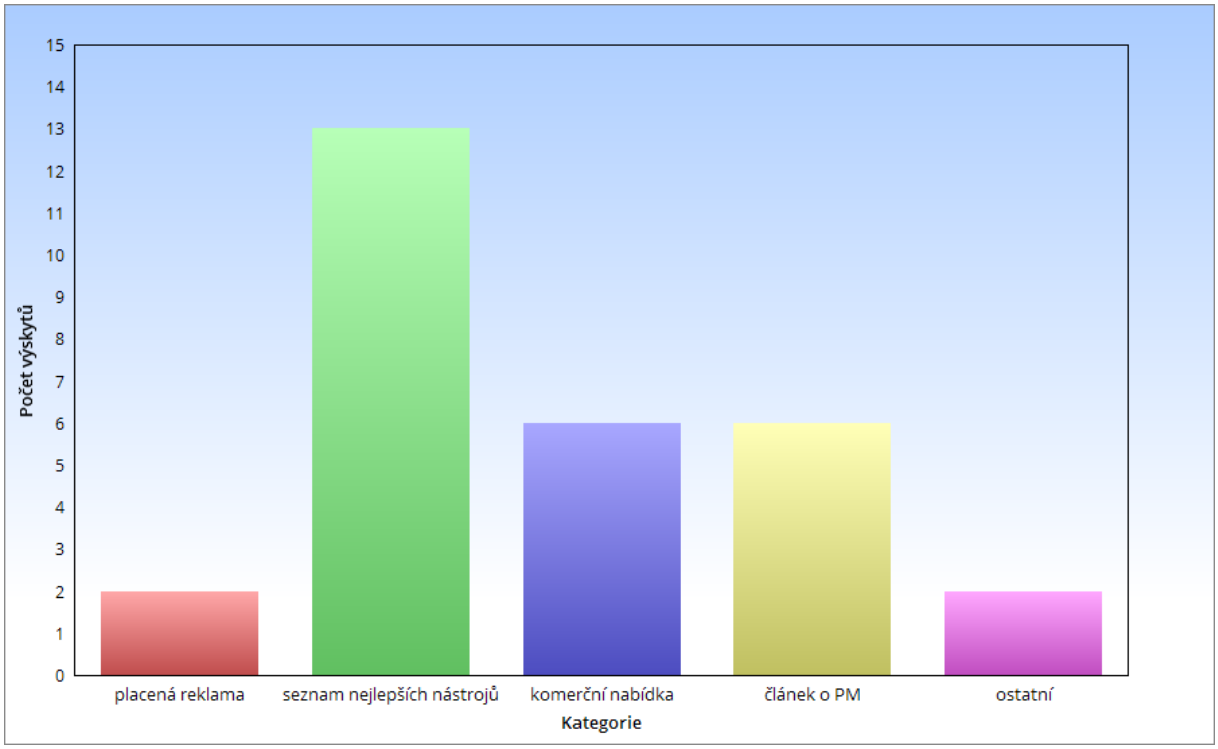
3.3 Výsledky vyhledávání pojmu Patch management software

V rámci této podkapitoly jsem provedl analýzu výsledků vyhledávání pojmu na internetu „*Patch management software*“.

Do analýzy jsem započítal prvních 30 nalezených výsledků, které jsem rozdělil na následující kategorie:

- placená reklama - reklamy softwarů pro podporu patch managementu poskytované společností Google
- seznam nejlepších nástrojů - články, které porovnávají komerční nabídky PM softwarů
- komerční nabídka - webové stránky výrobců patch managementu
- článek o PM - internetové články o obecné problematice patch managementu
- ostatní - články, které nesouvisí s patch managementem

Na obrázku 3.2 je uveden počet výskytů nalezených záznamů podle jednotlivých kategorií. Pro lepší pochopení jsou výsledná data reprezentována formou sloupového grafu.



Obrázek 3.2: Analýza vyhledávání pojmu Patch management software

Po analýze komerčních řešení, jsem zvolil pro detailnější zkoumání a porovnávání systémy, které splňují alespoň část základních požadavků. Abych získal potřebné informace o produktu, vždy jsem používal webové stránky výrobce.

Následující tabulka 3.2 ukazuje společné a odlišné funkčnosti vybraných systémů a jestli splňují základní požadavky zadavatele na systém (znázorněné tučným písmem). Ostatní vlastnosti systému jsem spolu s vedoucím servisního týmu vybíral podle jejich možného budoucího využití ve společnosti Corpus Solutions a.s.

Název produktu/ Vlastnost systému	Sledování endpointů	Testování patchů	Nasazení patchů	**Přidávání vlastních patchů**	**Podpora ITAM**	Podpora aplikací třetích stran
Manage Engine Patch Manager Plus [15]	ANO	ANO	ANO	ANO	NE	NE
Itarian Patch Management [16]	NE	NE	ANO	NE	ANO	NE
Solar Wings Patch Manager [17]	NE	ANO	ANO	NE	ANO	NE
Keseya VSA Patch Management [18]	ANO	NE	ANO	ANO	NE	NE

Tabulka 3.2: Porovnání komerčních nabídek pro Patch Management

Vysvětlení jednotlivých sloupců tabulky:

- Sledování endpointů - systém je schopen sledovat stav patchů nasazených na předem definovaných endpointech.
- Testování patchů - systém umožňuje uživateli nadefinovat pravidla pro automatické/manuální testování patchů.
- Nasazení patchů - systém je schopen automaticky nasazovat patche na assety.
- Přidávání vlastních patchů - systém umožní přidávat, aktualizovat patche, které si pak uživatel bude moci spojit s assetem.
- Podpora ITAM - systém je schopen spravovat assety.
- Podpora aplikací třetích stran - systém dokáže sledovat patche u aplikací třetích stran, jež používá servisní tým ve společnosti Corpus Solutions a.s. Seznam aplikací je možné najít na přiloženém CD.

Jak je vidět z tabulky, žádný ze zkoumaných mnou systémů nesplňuje základní požadavky (podkapitola 2.1), a proto není možné je použit v Coupus Solutions a.s.

3.4 Výsledky porovnání

Po konzultaci s ředitelem servisního týmu na základě vyhodnocení tabulek jsme došli k závěru, že žádná z nalezených aplikací nám nevyhovuje, a budu vyvíjet vlastní aplikaci.

Kapitola 4

Analýza procesu zadavatele

V této kapitole jsem popsal klíčový proces zadavatele Patch management (obrázky 4.1, 4.2), který poslouží základem mé aplikace. Dále jsem upřesnil funkcionalitu budoucího systému formou funkčních a nefunkčních požadavků.

4.1 Pojmy v procesu

V této podkapitole jsem vydefinoval pojmy, které jsou nezbytné pro pochopení procesu zadavatele, požadavků a samotnou implementaci systému.

SW na assetu

SW na assetu je softwarový produkt nasazený na konkrétním assetu. Může jít o operační systém, nebo dedikovaný softwarový produkt.

Hlavní větev produktu

Je hlavní vývojová větev sledovaného produktu, obvykle zavádí velké systémové a funkční změny. Nasazení takové verze realizuje obvykle projektový tým.

Podvětev / verze hlavní větve

Jde o podstatné funkční změny hlavní větve. Nasazení realizuje projektový nebo servisní tým.

Service Pack / Hotfix aklumulátor

Jedná se o větší opravu konkrétní podvětve implementující větší množství oprav, obvykle řešících problémy se stabilitou nebo výkonností. Nasazení realizuje servisní tým po konzultaci s TSA.

Patch-Fix

Jedná se o opravu konkrétního problému. Nasazení realizuje člen servisního týmu.

Kritický fix

Jedná se o opravu uvolněnou mimo běžnou roadmapu opravující konkrétní problém, obvykle bezpečnostní nebo s dopadem na stabilitu. Takový fix je nutné nasadit co nejdříve, realizuje člen servisního týmu.

4.2 Role v procesu zadavatele

Patch hunter

Je odpovědný za udržování aktuální databáze patch managementu. Sleduje webové stránky výrobce, release notes, e-maily, ostatní zdroje a v případě vydání nové verze nebo patche aktualizuje databázi patch managementu.

Patch tester

Je odpovědný za testování nově vydaného patche, buď obecně bez ohledu na plán nasazení u zákazníků, nebo v konkrétním případě zákaznické infrastruktury po konzultaci s TSA. Výsledky testování sdělí členům servisního a projektového týmu.

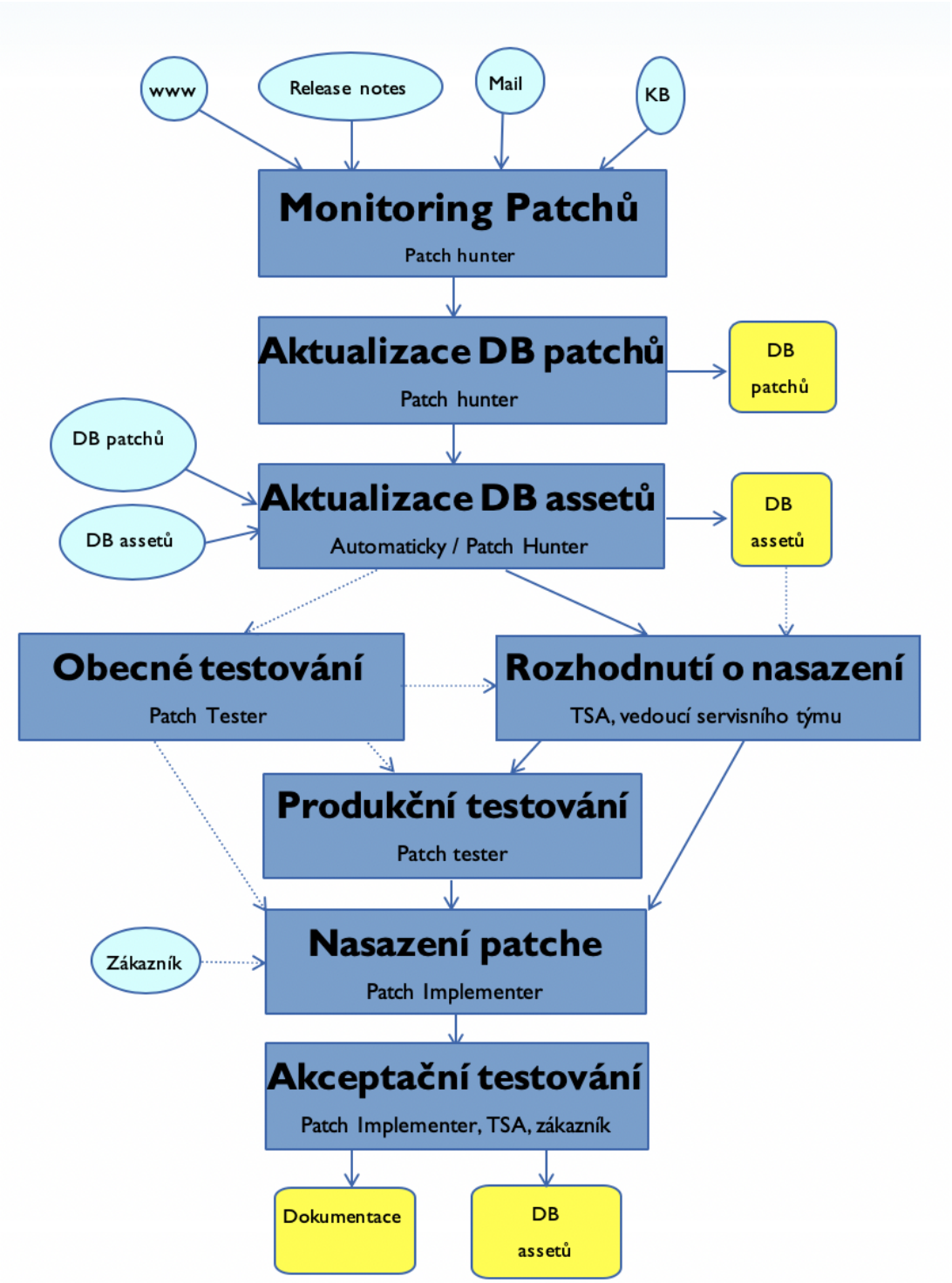
TSA

TSA (architekt technických řešení) každého zákazníka je zodpovědný za rozhodnutí o nasazení patche. Spolupracuje s patch testerem a po rozhodnutí o nasazení předává realizaci servisnímu nebo projektovému týmu.

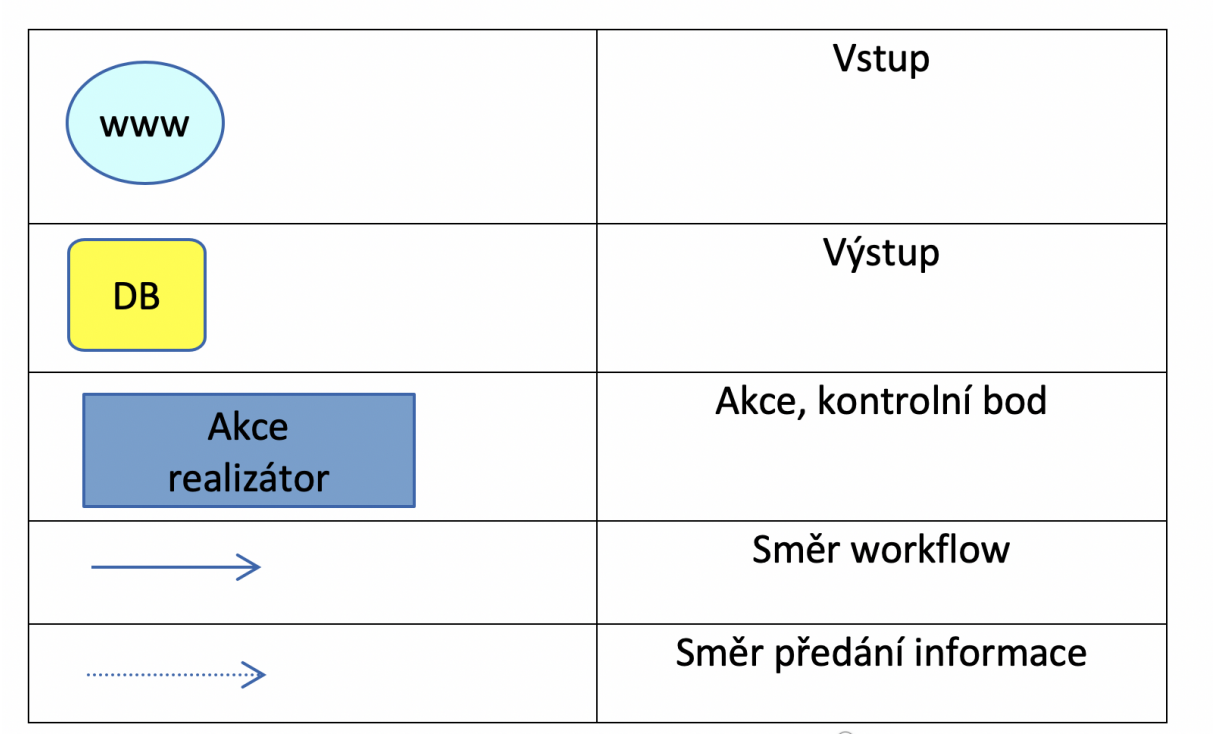
Patch implementer

Člen servisního nebo projektového týmu pověřený TSA nebo vedoucím týmu nasazením patche.

4.3 Proces patch management



Obrázek 4.1: Graf procesu patch management



Obrázek 4.2: Popis procesu patch management

Monitoring patchů

Patch hunter provádí pravidelné kontroly vydání nových patchů. Seznam lokací, kde a jakým způsobem lze dostupnost nových verzí zkontrolovat, je v databázi patchů.

Aktualizace databáze patchů

Po zjištění dostupnosti nového patche patch hunter aktualizuje databázi patchů a zapíše do ní datum aktualizace. Zároveň o dostupnosti nového patche informuje vedoucí servisního a projektového týmu.

Aktualizace databáze assetů

U každého assetu dojde k aktualizaci příslušného pole (tj. dostupné podvětve, Service Pack, Patch-Fix, kritický fix). K této aktualizaci by mělo dojít automatizovaně, srovnáním nasazené technologie a verze assetu, z databáze patchů. V případě dostupnosti nového patche systému asset managementu informuje TSA daného assetu, vedoucího servisního týmu a patch testera pro danou technologii.

Obecné testování

Pokud je vydaný patch většího rozsahu a týká se technologie, kde je pravděpodobné brzké nasazení u zákazníků (např. nová podvětev CheckPointu), je vhodné nejprve provést testování bez ohledu na testovací scénáře jednotlivých assetů. Výsledkem by měl být detailní přehled o vlastnostech nového patche, možná úskalí při implementaci a přehled nových funkcí.

Rozhodnutí o nasazení

Rozhodnutí o nasazení patche na konkrétní asset je v kompetenci TSA daného zákazníka. V případě drobných nebo kritických patchů může rozhodnutí provést vedoucí servisního týmu.

Výsledkem rozhodnutí o nasazení může být:

- Rozhodnutí patch nenasazovat a zapsání zdůvodnění do databáze assetů.
- Pokyn rolím patch tester a patch implementer k testování a nasazení patche.

Produkční testování

Jedná se o detailní testování patche v co nejpodobnějších reálných podmínkách. Od tohoto kroku může být rozhodnutím TSA nebo vedoucího servisního týmu upuštěno s ohledem na výsledky obecného testování a znalostí příslušné technologie.

Nasazení patche

Po domluvě se zákazníkem provede patch implementer nasazení patche.

Akceptační testování

Po nasazení patche se provede testování produkce buď podle testovacího scénáře zapsaného v databázi assetů, nebo podle domluvy se zákazníkem. Po akceptaci zákazníkem aktualizuje patch implementer databázi assetů a zkontroluje propagaci informací do interního dokumentačního systému.

4.4 Funkční požadavky

Po analýze procesu zadavatele následovala část rozšíření základních požadavků na systém (podkapitola 2.1). Požadavky jsem rozdělil do dvou kategorií: funkční a nefunkční.

4.4.1 Asset management

- **FP01 Správa skupin assetů**

  Systém umožní uživateli vytvářet a mazat skupiny assetů.

- **FP02 Správa assetů**

  Systém umožní uživateli vytvářet, editovat a mazat assety.

- **FP03 Vazba asset a skupina assetů**

  Systém bude schopen skupinám assetů přiřazovat jednotlivé assety.

4.4.2 Patch management

- **FP04 Správa SW**

  Systém umožní uživateli vytvářet, editovat, mazat SW.

- **FP05 Vazba asset a SW**

  Systém umožní uživateli přiřadit software k assetu.

- **FP06 Správa patchů**

  Systém umožní uživateli vytvářet, editovat, mazat patche v rámci softwarového produktu.

- **FP07 Zobrazení aktuálních verzí/patchů**

  Systém umožní uživateli vytvářet, editovat, mazat patche v rámci softwarového produktu.

- **FP08 Založit ticket do Jiry pro kontrolu vydání nového patche**

  Systém umožní pro každý softwarový produkt nastavit časovač, který určí jak často se provádí kontrola vydání nových patchů. Po vypršení časovače systém pro patch huntera konkrétního SW založí ticket do Jiry.

- **FP09 Založit ticket do Jiry pro kontrolu nového patche**

  Systém pro TSA nebo vedoucího servisního týmu založí ticket do Jiry pro rozhodnutí o nasazení nového patche.

- **FP10 Přihlášení pomocí Jiry**

  Systém umožní uživateli přihlašovat se pomocí Jiry.

4.5 Nefunkční požadavky

- **NP01 Výkonnost**

  Jelikož systém bude používat kolem 15 uživatelů (členů servisního týmu) denně, aplikace musí být schopna je obsloužit najednou.

- **NP02 Bezpečnost**

  Systém musí být dostatečně zabezpečený. To znamená, že musí být braněný oproti základním známým kybernetickým útokům (např. SQL Injection, Cross-site scripting).

- **NP03 Spolehlivost**

  Systém musí být dostatečně spolehlivý. To znamená, že a je schopen kdykoli poskytnout požadované služby v akceptovatelné kvalitě.

- **NP04 Uživatelská přívětivost**

  Systém musí být pro uživatele srozumitelný, jednoduchý a intuitivní na používání. To znamená, že ve stejný okamžik mu bude zobrazovat to, co uživatel potřebuje.

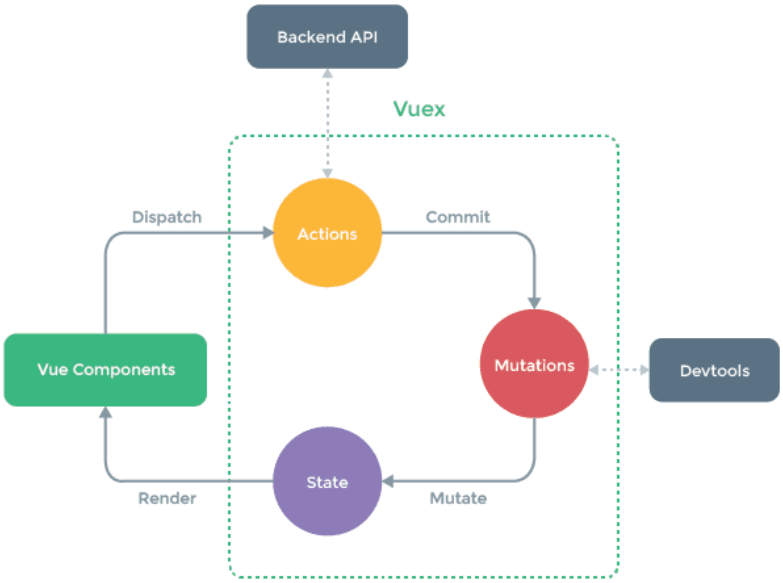
Kapitola 5

Návrh aplikace

Na základě analýzy procesu zadavatele (kapitola 4) byl v rámci této kapitoly proveden návrh aplikace. Definoval jsem architekturu, programovací jazyky a frameworky, které byly vybrány k realizaci této práce. Důležitost kapitoly spočívá v tom, že správně zvolené technologie přímo ovlivňují dobu a průběh implementace.

5.1 Architektura

Aplikace bude postavená na architektuře backend-frontend a REST API používaným jako rozhraní pro komunikaci mezi serverem a klientskou stranou. Backend strana se používá pro operace CRUD s daty, která jsou uložena v PostgreSQL databázi, komunikaci a autorizaci s Jirou. Frontend sestává z úložiště Vuex (což je implementace architektury Flux pro VueJS) a komponent pro každou entitu v této aplikaci. Následující diagram ukazuje vztahy mezi komponentami.



Obrázek 5.1: Architektura aplikaci[19]

5.2 Použité technologie

5.2.1 Front-end

Pro vytvoření klientské části aplikace jsem použil základní balíček technologií:

- HTML (HyperText Markup Language) - jednoduchý způsob, jak strukturovat obsah webové stránky.
- CSS (Cascading Style Sheets) popisuje, jak se mají prvky HTML zobrazovat na obrazovce.
- JavaScript přidává interaktivitu na stránku.

Na základě mých předchozích zkušeností s vývojem webových aplikaci jsem se rozhodl použít JavaScriptový framework, který mi pomůže psát kód efektivněji. Vybíral jsem mezi třemi nejpopulárnějšími frameworky pro JavaScript: Vue.js, React, Angular[20].

Následující tabulka 5.1 ukazuje kritéria, která byla použitá při výběru frontend frameworku, a hodnocení, kde 1 je nejlepší a 3 nejhorší.

Název produktu/ Vlastnost frameworku	**Angular**	**React**	**Vue.js**
Dokumentace	2	1	1
Snadno začít	3	2	1
Předchozí zkušenost	3	3	1
Popularita	2	1	2
Potřebné znalosti	TypeScript	JS, JSX	JS
Flux-like architektura	3	1	1

Tabulka 5.1: Porovnávání frontend frameworků

Porovnávací tabulka je založená na zkušenostech autora s výše zmíněnými frameworky a je subjektivní. Jako jedno z důležitých kritérií jsem zvolil přítomnost ve frameworku Flux-like architektury, která poskytuje operace s daty založenými na myšlence jednosměrného toku dat. Ve VueJS je oficiálně doporučená a podporovaná implementace Flux, která se jmenuje Vuex. Flux pattern umožňuje vytvářet nezávislé komponenty, které vždy představují skutečná data.

Na základě vyhodnocení v tabulce jsem se ve své aplikaci rozhodl použít framework VueJS.

Bootstrap Vue

Během jedné konzultace se zadavatelem jsem zjistil, že zaměstnanci Corpus Solutions a.s. používají interní systémy nejen z pracovního počítače, ale také z tabletu nebo telefonu. Proto jsem se nad rámec definovaných nefunkčních požadavků rozhodl, že moje aplikace bude podporovat různé typy zařízení. Pro řešení tohoto požadavku jsem použil technologii Bootstrap Vue.

Bootstrap je open-source CSS framework zaměřený na responzivní vývoj webových aplikací. Obsahuje před-připravené šablony založené na CSS a JavaScript pro formuláře, tlačítka, navigaci a další komponenty rozhraní. Bootstrap Vue je knihovna, která zabaluje Bootstrap 4 kolem Vue.js a usnadňuje vytváření komponent[21].

5.2.2 Back-end

Pro napsání serverové části aplikace jsem se rozhodl použít Node.js.

Node.js je javascript prostředí, jehož základem je Chrome V8 engine. Tento engine převádí kód JavaScript na rychlejší strojový kód. Také Node.js používá událostně řízený, neblokující I / O model, díky němuž aplikace může obsloužit mnoho připojených klientů najednou. Další výhodou je to, že ekosystém balíčku Node.js npm je největší ekosystém knihoven s otevřeným zdrojovým kódem na světě[22].

LoopBack

Frameworky Node.js se většinou používají kvůli jejich produktivitě, škálovatelnosti a rychlosti, a usnadňují vytváření enterprise aplikací. Udělal jsem průzkum nejpopulárnějších z nich a vybral jsem ten, který pro můj systém nejvíce vyhovoval.

LoopBack od společnosti StrongLoop (IBM) je vysoce rozšiřitelná, open-source platforma Node.js, která umožňuje jednoduše vytvářet dynamická koncová REST API.
Loopback umožňuje vytvářet modely na základě schématu nebo dynamických modelů bez schématu. Je kompatibilní s velkým množstvím REST API služeb a celou řadou databází včetně MySQL, Oracle, MongoDB, Postgres atd. Framework poskytuje také oddělitelné komponenty pro ukládání souborů, přihlášení třetích stran a OAuth 2.0[23].

5.2.3 Databáze

Pro ukládání dat aplikace jsem zvolil databázi PostgreSQL[24].

PostgreSQL je výkonný, open source objektově relační databázový systém, který používá a rozšiřuje jazyk SQL v kombinaci s mnoha funkcemi, které bezpečně ukládají velké množství dat. PostgreSQL má mnoho pokročilých funkcí jako například:

- Uživatelem definované typy
- Dědičnost tabulek
- Propracovaný blokovací mechanismus
- Referenční integrita cizího klíče
- Pohledy, pravidla, poddotaz
- Vnořené transakce
- Ovládání souběžnosti více verzí (MVCC)
- Asynchronní replikace

Další výhodou vybrané databázové technologie je to, že většina interních systémů Corpus Solutions a.s. ji také používá a každý měsíc se provádí zálohování dat ze všech Postgre databází.

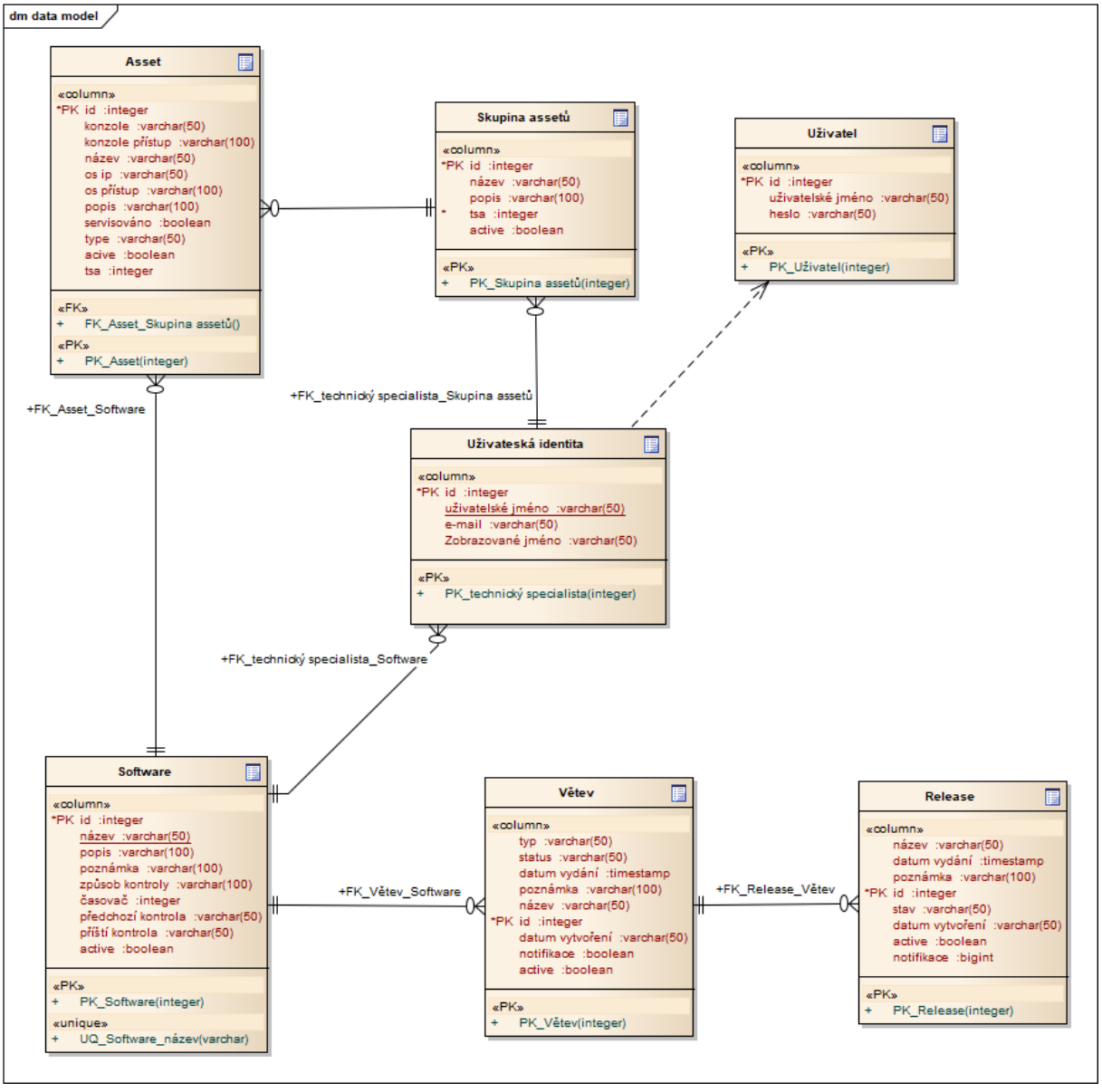
5.3 Diagramy

Pro vizuální reprezentaci návrhu řešení jsem se rozhodl použít modelovací jazyk Unified Modeling Language (UML), který se skládá z integrované sady diagramů vyvinutých pro specifikaci, vizualizaci, konstrukci a dokumentování artefaktů softwarových systémů. UML představuje soubor nejlepších technických postupů, které se osvědčily při modelování velkých a složitých systémů, a je velmi důležitou součástí procesu vývoje softwaru[25].

Na základě funkčních požadavků (viz podkapitola 2.1) a navržené architektury jsem vytvořil digram tříd a digram případů užití (obrázky 5.2, 5.3).

5.3.1 Diagram tříd

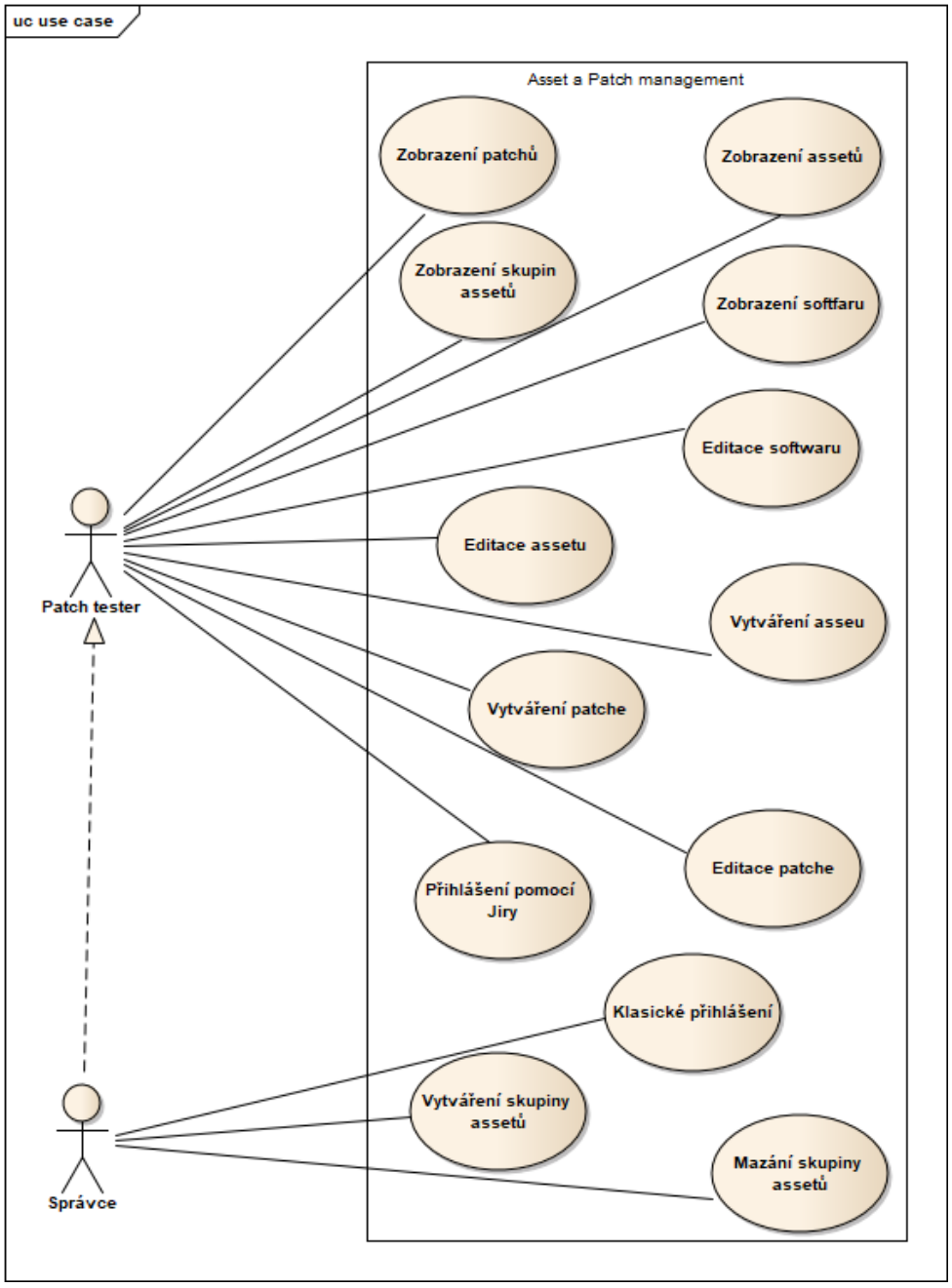
V UML diagram tříd je typ statického diagramu, který popisuje strukturu systému tím, že zobrazuje třídy systému, jejich atributy, metody a vztahy mezi objekty.



Obrázek 5.2: Diagram tříd

5.3.2 Diagram případů užití

Diagram případů užití se obvykle označuje jako diagram chování používaný k popisu souboru akcí (případy užití), které systém může provádět ve spolupráci s jedním nebo více externími uživateli systému (aktéry). Případy užití umožňují spojit to, co potřebujeme od systému, s tím, jak systém tyto potřeby plní.[25]



Obrázek 5.3: Diagram případů užití

Kapitola 6

Implementace

V této kapitole se budu věnovat průběhu implementace systému. Nejprve popíšu nástroje, které jsem pro implementaci použil. Poté se budu věnovat uživatelskému rozhraní aplikace a klientské části. Nakonec popíšu způsoby a metodiky řešení některých požadavků.

6.1 Použité nástroje

Visual Studio Code

Visual Studio Code je editor vyvinutý společností Microsoft. Zahrnuje podporu pro debugování, vestavěné řízení Git a GitHub, zvýrazňování syntaxe, inteligentní dokončení kódu, snippety a refaktoring kódu. Je vysoce přizpůsobitelný a umožňuje uživatelům měnit téma, klávesové zkratky, předvolby a instalovat rozšíření, která přidávají další funkce[26].

Visual Studio Code jsem použil jak pro napsání frontendové, tak i backendové části a také pro formátování JSON souborů.

pgAdmin

pgAdmin je Open Source nástroj pro správu PostgreSQL. pgAdmin 4 poskytuje výkonné grafické rozhraní, které zjednodušuje vytváření, údržbu a používání databázových objektů. Také nástroj umožňuje jednoduše psát dotazy a exportovat výsledky do csv souboru pro lepší zkoumání dat[27].

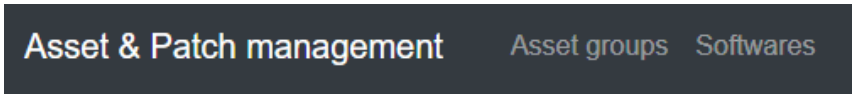
6.2 Uživatelské rozhraní

Frontendová část je implementována na základě principů Single Page Application (SPA). SPA je moderní způsob vytváření webové aplikace, v němž si uživatel jednou načte HTML stránku a poté s ní interaguje, zatímco stránka dynamicky přepíše svůj obsah. Takový přístup umožňuje zabránit přerušení UI mezi načítáním stránek a vytvoří aplikaci tak, aby vypadala jako nativní. To znamená, že si uživatel jednou stáhne veškerý obsah stránky (komponenty / grafické prvky uživatelského rozhraní) a poté komunikuje se serverem pouze za účelem odeslání nebo načtení určitých dat přes backend API.

Jedním z nefunkčních požadavků byla vysoká uživatelská přívětivost systému (viz podkapitola 4.5). Během návrhu a implementace jsem četl články o tom, jak správně a lépe navrhnout UI/UX.

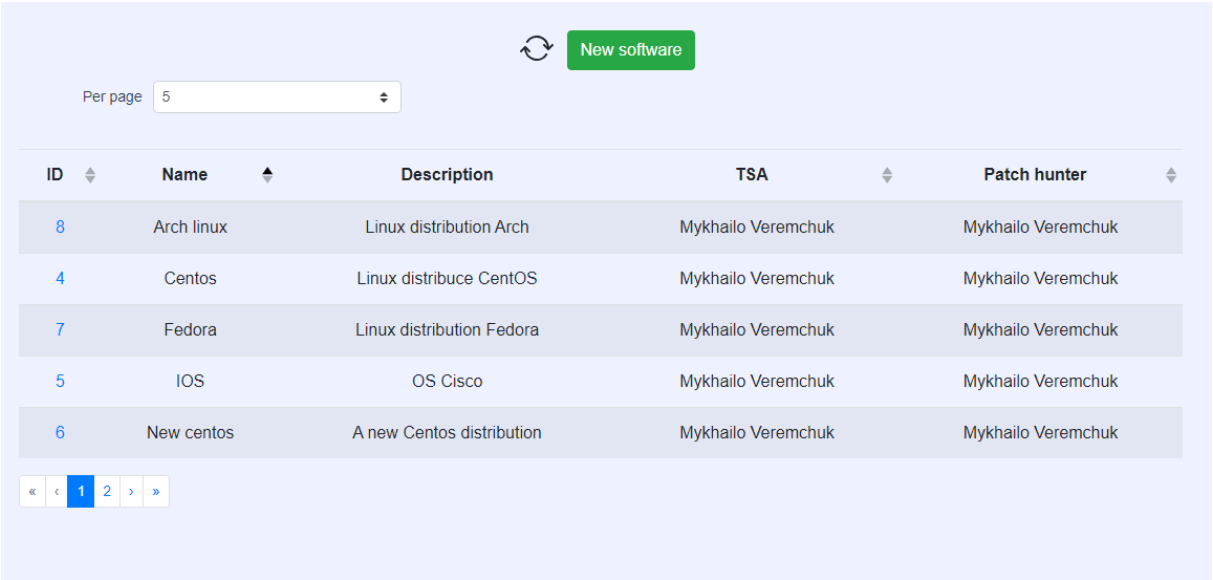
Po načtení aplikace se zobrazí přihlašovací formulář (obrázek 6.11). Po přihlášení v navigační liště jsou 2 sekce (obrázek 6.1):

- Asset groups - seznam všech skupin assetů
- Softwares - číselník sledovaných softwarových produktů

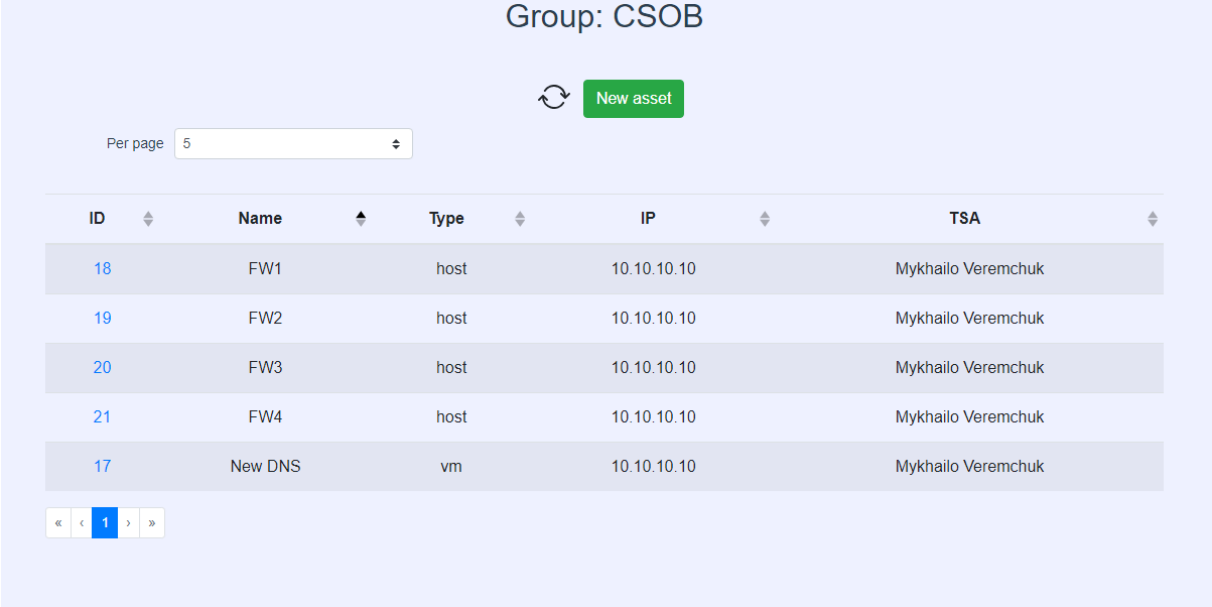


Obrázek 6.1: Navigační lišta

V aplikaci jsou seznamy skupin assetů, assetů, softwarových produktů, větví a patchů prezentovány formou tabulky, která obsahuje nejdůležitější údaje jednotlivých položek. Tabulky lze seřadit podle jakéhokoliv povinného atributu. Také uživatel může změnit počet zobrazených položek na stránku, obnovit tabulku a přidat nový záznam (obrázky 6.2, 6.3).

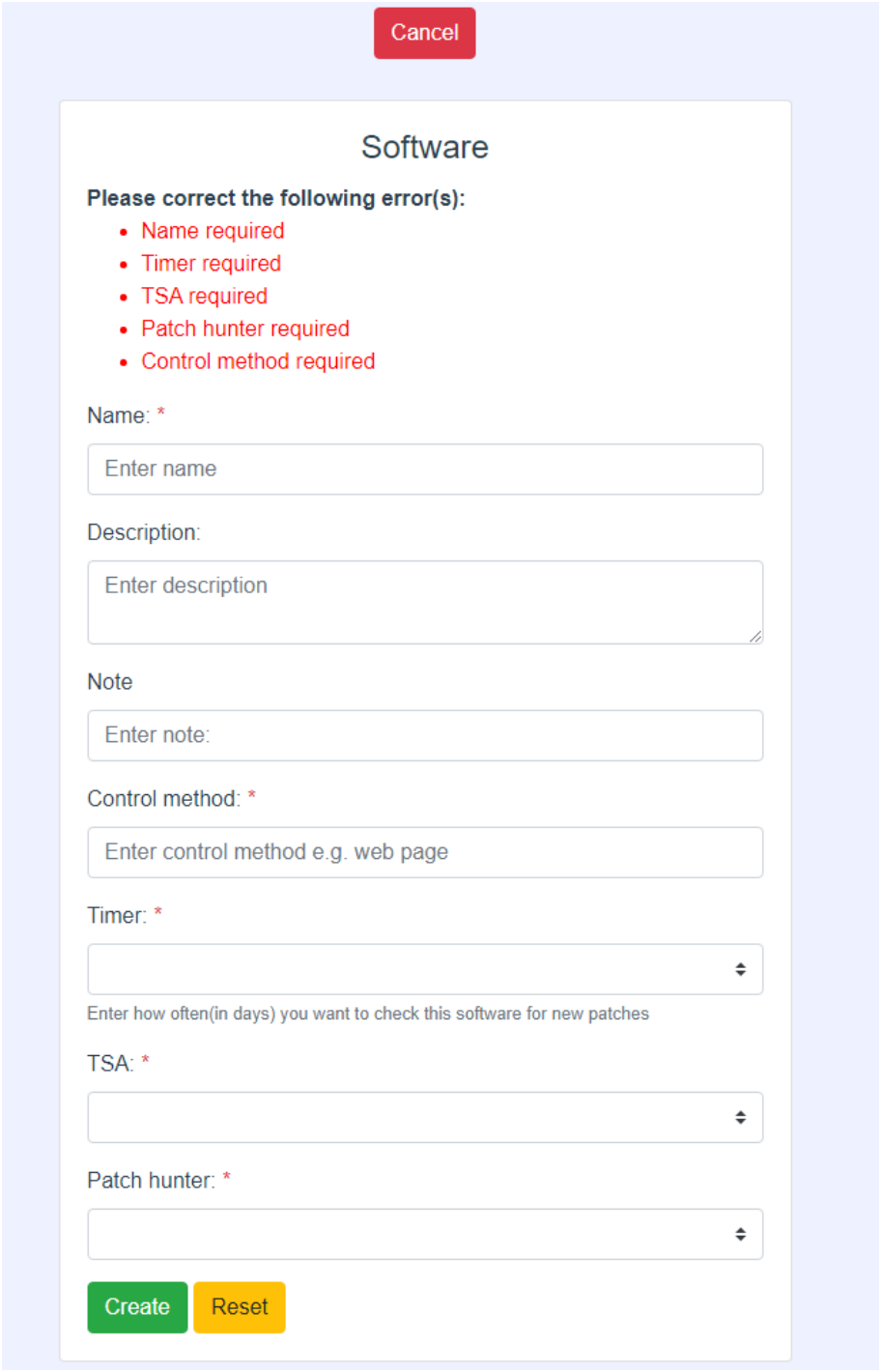


Obrázek 6.2: Seznam všech softwarových produktů

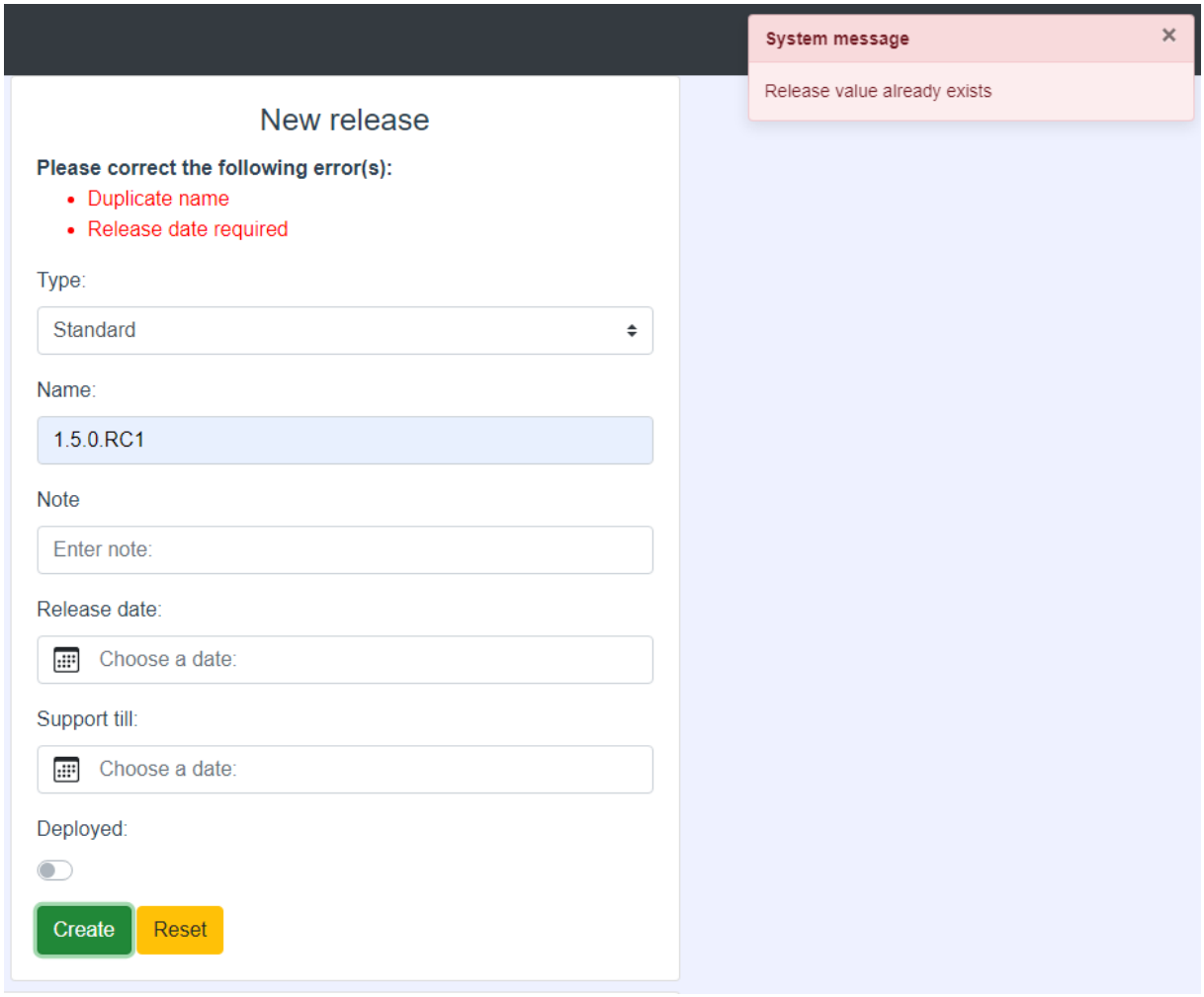


Obrázek 6.3: Seznam assetů vybrané skupiny assetů

Na všech formulářích funguje validace na vyplnění povinných položek a kontrola duplicity názvu (obrázky 6.4, 6.5).

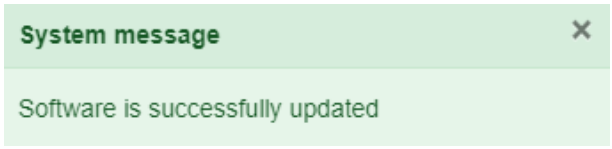


Obrázek 6.4: Formulář vytvoření nového softwaru

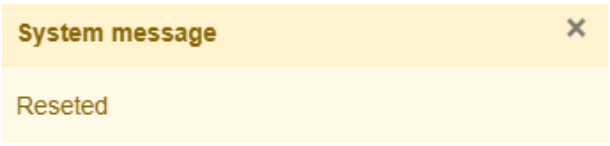


Obrázek 6.5: Formulář vytvoření nového patche

Pro lepší upozornění uživatele v aplikaci jsou použity krátké vyskakovací zprávy (obrázky 6.6, 6.7).

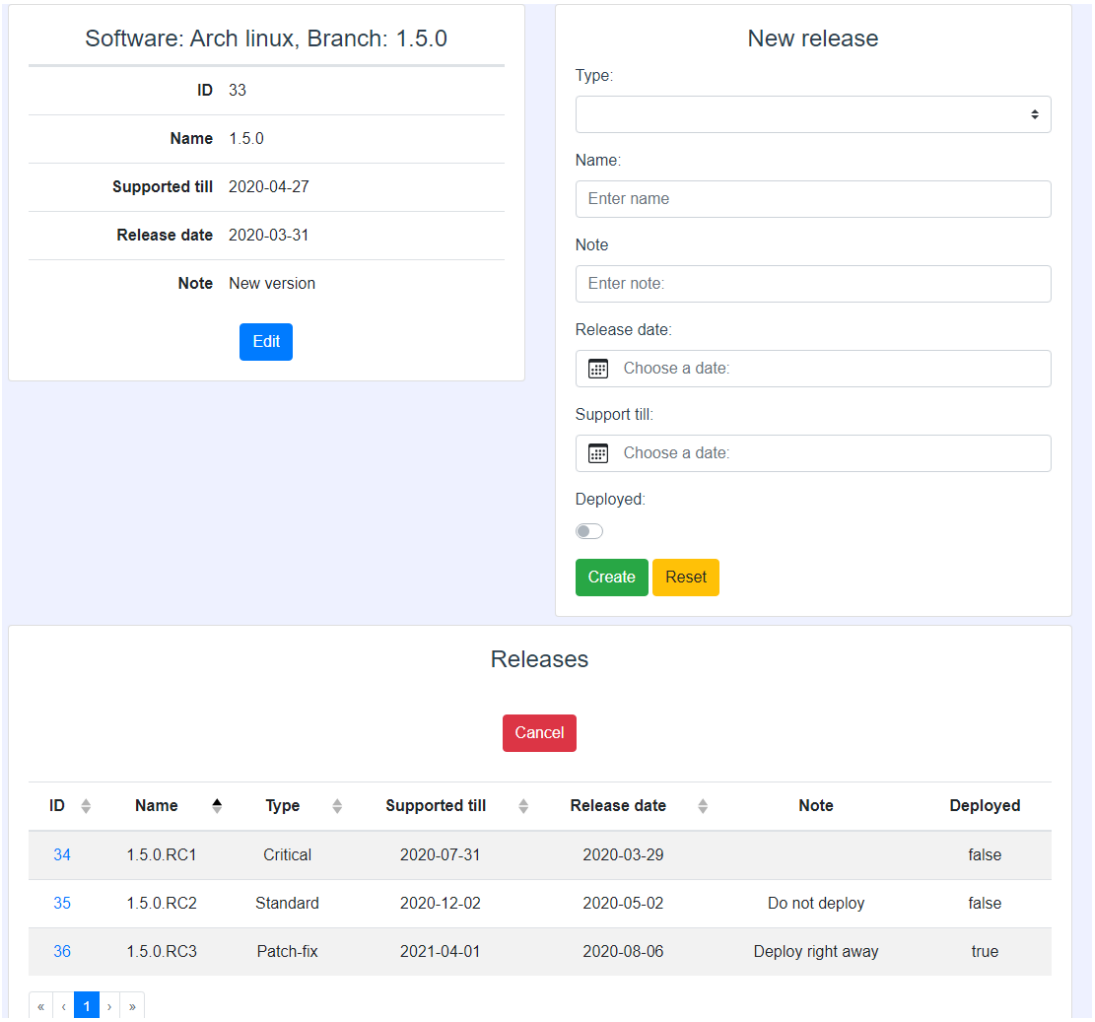


Obrázek 6.6: Upozornění o aktualizaci

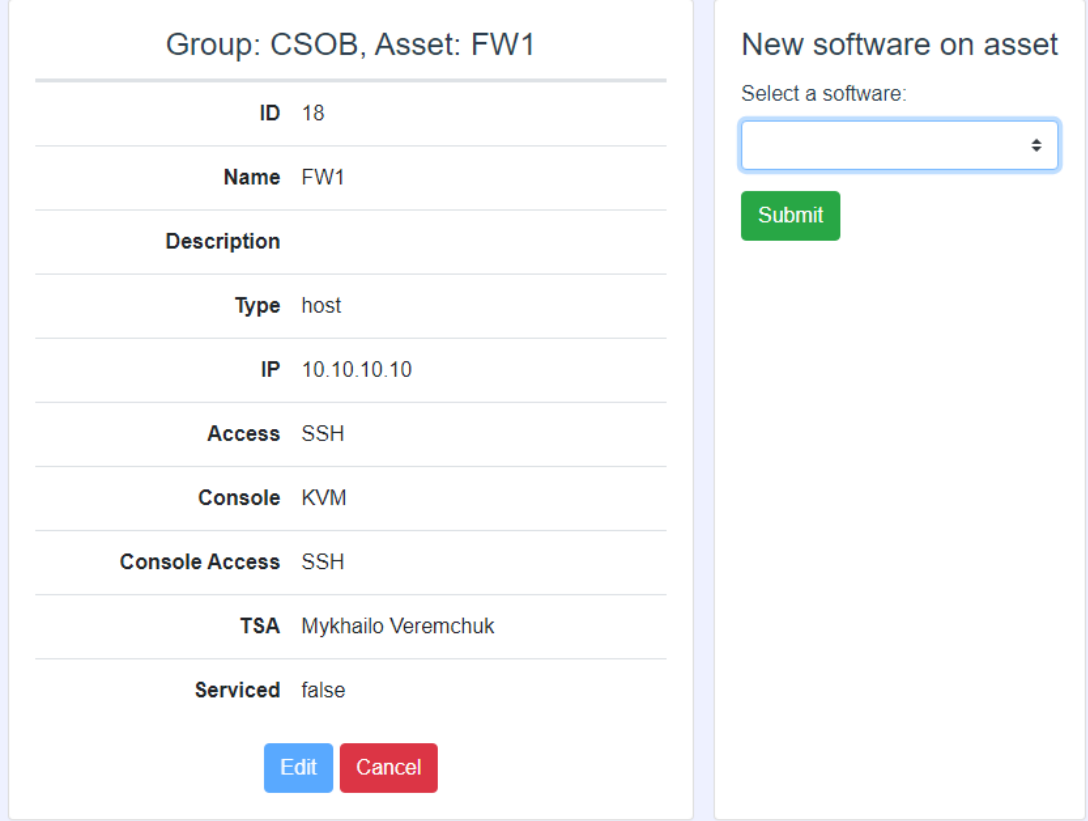


Obrázek 6.7: Upozornění o obnoveni

Po kliknutí na jakýkoliv prvek tabulky: asset, software, větev, patch se zobrazí stránka se všemi atributy vybrané položky. Také uživatel si bude moci upravit nebo smazat prvek, případně provést další akce (obrázky 6.8, 6.9).

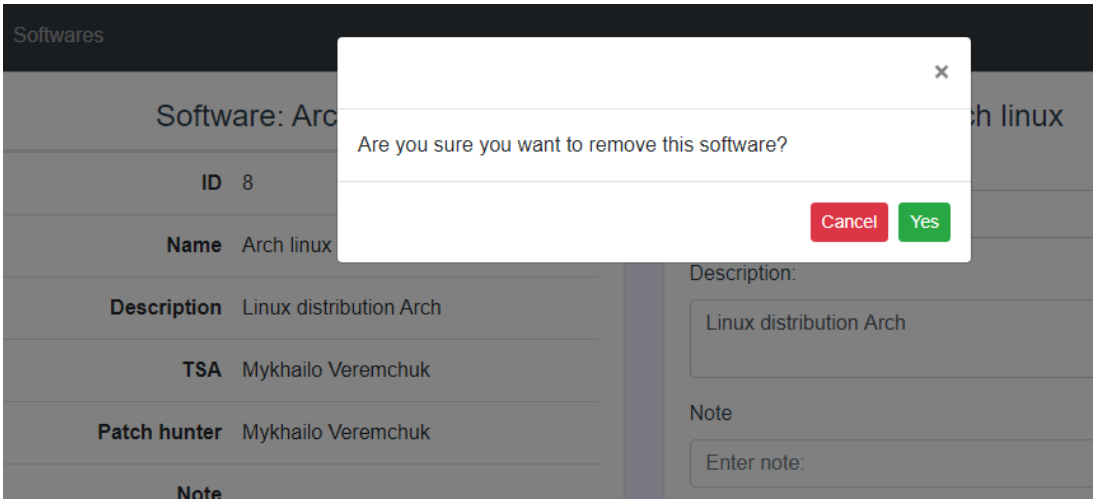


Obrázek 6.8: Detail větve softwaru s možností přidání nového patche



Obrázek 6.9: Detail assetu s možností přidání softwaru

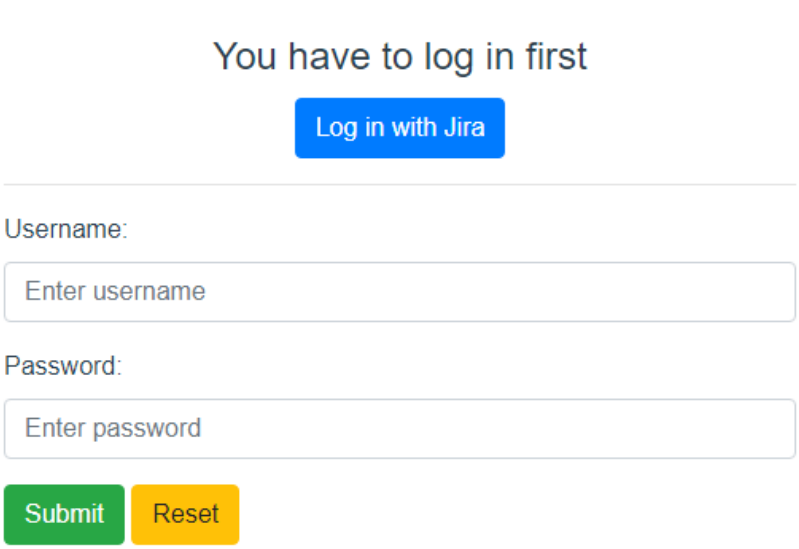
Při pokusu o smazání jakéhokoliv prvku se zobrazí potvrzovací modální okno (obrázek 6.10).



Obrázek 6.10: Potvrzovací modální okno

6.3 Zabezpečení a přihlášení

Jedním z nefunkčních požadavků je vysoká zabezpečenost systému (viz podkapitola 4.5). Jelikož aplikace řídí přístup k datovým a volacím službám, je nutné vytvořené REST API patřičně zabezpečit. Chráněné údaje musí být přístupné pouze pro přihlášené uživatele. Pro přihlášení admina a testování jsem použil metodu frameworku loopback která poskytuje vnořenou autentizaci pomocí tokenů. Pro běžné uživatele tento způsob autentizace je nedostatečný, jelikož jedním z požadavků na systém je propojenost aplikací s Jirou (viz podkapitola 4.4). Pro řešení tohoto požadavku jsem použil technologii OAuth 1.0a.



Obrázek 6.11: Přihlašovací formulář

6.3.1 OAuth 1.0a

OAuth je otevřený standard pro delegování přístupu, který se běžně používá jako způsob, jak uživatelé udělí aplikacím přístup k jejich informacím na jiných webových stránkách, aniž by jim však poskytli hesla. Místo toho OAuth1 používá autorizační tokeny s jednosměrným kryptografickým podpisem HMAC-SHA1 k prokázání identity mezi spotřebitelem a poskytovatelem služeb. OAuth1 je široce používaný, testovaný, bezpečný protokol, který lze použít i bez SSL[28].

V praxi to znamená, že po kliknutí na tlačítko Log in with Jira (obrázek 6.11) se načte přihlašovací stránka Jiry, kam uživatel zadá své údaje, a je zpátky přesměrován do aplikace. V případě, že je uživatel v tomto prohlížeči již přihlášen do Jiry po kliknutí na přihlašovací tlačítko, musí pouze souhlasit s poskytnutím svých údajů (bez hesla) této aplikaci.

6.4 Zakládání požadavků do Jiry

Často je v aplikaci třeba spouštět nějaké naplánované úlohy jako např. každodenní zálohování dat, odesílání e-mailů atd. Pro tyto učely lze využit externí knihovny (např. node-cron). Cron je časový plánovač úloh, který umožňuje aplikacím naplánovat automatické spuštění úlohy na konkrétní datum nebo čas nastavením cron formatu (obrázek 6.12).

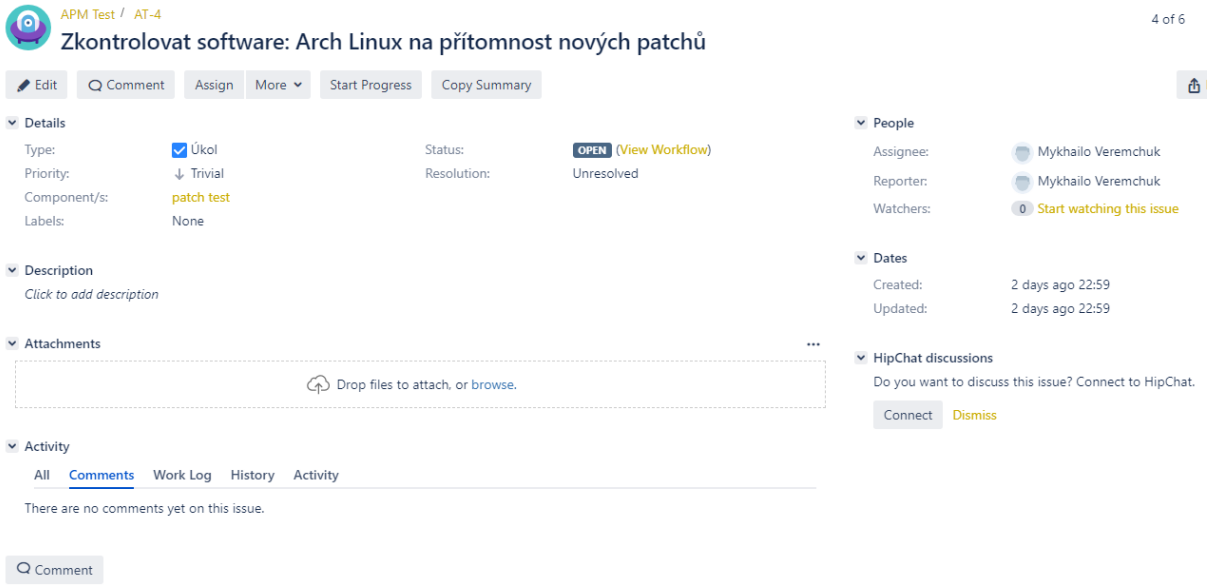
```
# ┌──────────── min (0 - 59)
# │ ┌────────────── hour (0 - 23)
# │ │ ┌──────────────── day of month (1 - 31)
# │ │ │ ┌────────────────── month (1 - 12)
# │ │ │ │ ┌──────────────────── day of week (0 - 6) (0 to 6 are Sunday to
# │ │ │ │ │                     Saturday, or use names; 7 is also Sunday)
# │ │ │ │ │
# │ │ │ │ │
# * * * * *  command to execute
```

Obrázek 6.12: Cron format[29]

Tuto technologii jsem se rozhodl použít pro implementaci požadavků na zakládání ticketů do Jiry (viz podkapitola 4.4).

```
cron({
    on: '0 0 * * *'
}, function() {
    var today = new Date();
    var tomorrow = new Date();
    tomorrow.setDate(today.getDate() + 1);
    app.models.software.find({
        where: {
            active: true,
            nextControl: {
                lte: tomorrow
            }
        },
        include: {
            relation: 'TSA'
        }
    }, function(err, returned) {
        if (err) console.log(err);
        else {
            if (returned !== null && returned.length !== 0) {
                returned.forEach(function(software) {
                    createIssue(software, null, null);
                });
            }
        }
    });
});
```

Tato funkce pro patch huntery zakládá požadavky na kontrolu vydání nových patchů a je nastavena tak, aby se pouštěla každý den o půlnoci. Toto nastavení zaručí řádek: 0 0 * * *. Nejdřív si backend stáhne všechny softwarové produkty, u nichž za jeden den vyprší časovač kontroly. Dále se pro každý SW zavolá funkce createIssue, která vytvoří JSON objekt a pošle ho do API Jiry. Tím se vytvoří požadavek na kontrolu nových patchů (obrázek 6.13).



Obrázek 6.13: Nový požadavek na kontrolu nových patchů

Kapitola 7

Testování

Před nasazením do reálného prostředí musí být funkčnost systému pořádně otestována. Testování aplikace jsem rozdělil do dvou částí: funkční a uživatelské testy.

Funkční testování je typ testování softwaru, který ověřuje softwarový systém podle funkčních požadavků / specifikací. Účelem funkčních testů je otestovat každou funkci softwarové aplikace. Funkce jsou testovány poskytnutím vhodného vstupu a ověření výstupu. Tento typ testování kontroluje uživatelské rozhraní, rozhraní API, databázi, zabezpečení, komunikaci mezi klientem a serverem a další funkce testované aplikace[30]. Funkční testování jsem prováděl po realizaci každého funkčního požadavku.

7.1 Uživatelské testování

Uživatelské testování je speciální technika, která umožňuje vyhodnotit použitelnost systému. Během uživatelského testování je účastníkům představen systém, pak se účastníci s ním snaží pracovat a nakonec vyplní speciální formulář, kde jsou popsány jejich zkušenosti se systémem. Tento způsob testování je důležitý, protože nám umožňuje sledovat, jak potenciální uživatel bude aplikaci používat.

7.1.1 Příprava

Pro testery jsem připravil 3 testovací scénáře, které se skládají z jednotlivých případů užití a pokrývají veškerou funkcionalitu aplikace. Případy užití obsahují informaci o tom, jakou akci má uživatel provést bez popisu jednotlivých kroků. Timto způsobem se ověří srozumitelnost, jednoduchost a intuitivnost aplikace.

Scénář A

- Přihlásit se jako správce pomocí těchto údajů admin heslo
- Vytvořit novou skupinu assetů
- Pro nově vytvořenou skupinu assetů přidat nový asset
- Upravit některé položky assetu
- Provázat asset s již existujícím softwarem

- Změnit software na assetu
- Odebrat software z assetu
- Smazat asset
- Odhlásit se

Scénář B

- Přihlásit se pomocí Jiry
- Seřadit tabulku softwarů podle patch huntera
- Změnit počet zobrazených položek na stránku
- Ověřit refresh tabulky vytvořením softwaru v jiném tabu a kliknutím refresh tlačítka v původním
- Upravit některé položky softwaru
- Ověřit reset editačních položek softwaru
- Provázat již existující asset s novým softwarem
- Odhlásit se
- Ověřit založení požadavku pro kontrolu vydání nových patchů do Jiry

Scénář C

- Přihlásit se pomocí Jiry
- Vytvořit nový software
- Na softwaru vytvořit novou větev
- Upravit některé položky větve
- Smazat větev
- Na softwaru vytvořit novou větev
- Na větvi vytvořit nový release
- Upravit některé položky releasu
- Smazat release
- Odhlásit se
- Ověřit založení požadavku pro kontrolu jedné větve do Jiry

Uživatelského testování se zúčastnili celkem čtyři lidé. Nejprve byl uživateli představen krátký popis aplikace a základní funkčnost. Dále se uživatel seznámil s testovacími scénáři a přistoupil k jejich realizaci. Já jsem jako pozorovatel měl možnost sledovat, na kterém úkolu účastník strávil nejvíc času. Po skončení testu byl účastníkovi předložen krátký dotazník k vyplnění. Dotazník sestával z řady otázek zaměřených na získání zpětné vazby od účastníků.

Příklad otázek:

- S jakými problémy jste se během testu setkali?
- Která část testu byla pro vás nejobtížnější a proč?
- Co byste v aplikaci zlepšili?

7.1.2 Výsledky testování

- Účastník 1 - student ČVUT, v současné době pracuje jako frontend vývojář.

  Scénář A - Během testu se nevyskytl žádný problém.

  Scénář B - Během testu se nevyskytl žádný problém.

  Scénář C - Během testu se nevyskytl žádný problém.

  Poznámky: Aplikace vypadá dobře, líbí se mi validace formulářů a vyskakovací zprávy.

- Účastník 2 - absolvent ČVUT, v současné době pracuje jako android vývojář.

  Scénář A - Během testu se nevyskytl žádný problém.

  Scénář B - Během testu se nevyskytl žádný problém.

  Scénář C - Během testu se nevyskytl žádný problém, ale některá pole formuláře nejsou srozumitelná.

  Poznámky: Může být vhodné přidat osobní stránku, na které bude mít uživatel přehled o svých softwarových produktech a naplánovaných kontrolách.

- Účastník 3 - absolvent ČVUT, v současné době pracuje jako technický specialista pro Corpus Solutions a.s.

  Scénář A - Během testu došlo k některým problémům: po vytvoření se asset v tabulce neobjevil. Musel jsem kliknout na tlačítko obnovit.

  Scénář B - Během testu se nevyskytl žádný problém.

  Scénář C - Během testu se nevyskytl žádný problém.

  Poznámky: Chybělo mi tlačítko které by mě vrátilo na předchozí objekt.

- Účastník 4 - student ČVUT.

  Scénář A - Během testu se nevyskytl žádný problém.

  Scénář B - Během testu se nevyskytl žádný problém.

  Scénář C - Během testu se nevyskytl žádný problém.

Kapitola 8

Budoucí rozvoj aplikace

Po domluvě s externím zadavatelem se rozhodlo, že tento projekt budu mít na starosti i po odevzdání bakalářské práci. Do interní Jiry se založí nový projekt, kam uživatelé aplikací budou moci zakládat požadavky na změnu. Bez ohledu na to, že je aplikace použitelná a plně funkční, lze ji rozšířit o další funkčnost.

Během testování aplikace zadavatelem vznikly následující požadavky na změnu:

- Přemístit aplikaci do prostoru Confluence.

  To znamená, že místo skupin assetů se pro každého zákazníka v Confluence založí zvláštní prostor, po jehož otevření se načtou jeho assety. Tato změna zabrání rozšíření počtu interních systémů ve firmě a zjednoduší nastavování práv pro jednotlivé uživatele.

- Vytvořit zjednodušené přehledy o nasazení patchů pro manažery.

  Toto vylepšení pomůže manažerům se rychle zorientovat v aktuálním stavu zákaznických assetů bez potřeby se doptávat technika.

- Vylepšení uživatelského rozhraní.

Na zavěrečné konzultaci se zadavatelem se rozhodlo, že aplikace bude nasazená do provozu až po implementaci těchto požadavků a připomínek účastníků uživatelského testování (viz podkapitola 7.1.2).

Závěr

Cílem této práce bylo provést analýzu požadavků externího zadavatele a následně navrhnout a implementovat systém. Dalším cílem bylo ověřit funkčnost aplikace formou uživatelských testů.

V rámci bakalářské práce jsem zjistil potřeby zadavatele, detailněji analyzoval jeho procesy a určil požadavky na systém. Následně byla navržena architektura a uživatelské rozhraní aplikace. Pro lepší vizuální reprezentaci návrhu řešení byly také vytvořené UML diagramy. Všechny požadavky uvedené v analýze byly implementovány spolu s několika novými funkcemi, které nebyly součástí specifikace požadavků. Po dokončení implementace byly provedeny uživatelské testy, které pomohly určit požadavky pro budoucí rozvoj. Nezbytnou součástí práce bylo seznámení s obecnou problematikou Patch a Asset managementu, prozkoumání a vyhodnocení existujících řešení.

Výhodou navrženého systému v porovnání se stávajícím je vyšší spolehlivost, lepší uživatelské rozhraní a propojení s interními aplikacemi. Systém byl vyvinut podle klíčového procesu ve společnosti. Nový způsob evidence zákaznických assetů a softwarových produktů ušetří servisnímu týmu Corpus Solutions a.s. několik hodin měsíčně, jež mohou věnovat důležitějším věcem. Díky tomu bude očekávaným přínosem i lepší komunikace technických specialistů se svými zákazníky.

Výstupem práce je funkční aplikace v alfa verzi která se bude dále rozvíjet a vylepšovat. Všechny stanovené cíle byly splněny, a práci považuji za úspěšnou. Věřím, že pro Corpus Solutions a.s. bude moje aplikace přínosná a těším se na další spolupráci. Pro mě osobně to byl první větší softwarový projekt. Prošel jsem životním cyklem vývoje informačního systému od stanovení business požadavků po implementaci a testování.

Příloha A

Slovník pojmů

Frontend - prezentační vrstva aplikace

Endpoint - vzdálené výpočetní zařízení, které komunikuje tam a zpět se sítí, ke které je připojeno

SQL Injection - typ útoku na databázovou vrstvu přes neošetřený vstup

Cross-site scripting - typ útoku na webovou stránku pomocí škodlivého skriptu přes neošetřený vstup

Příloha B

Struktura CD

```
/
├── client
│   ├── dist
│   └── src
├── server
│   ├── boot
│   ├── common
│   ├── package.json
│   ├── middelware.json
│   ├── config.json
│   └── server.js
├── screenshots
├── softwareProducts.txt
└── thesis
    ├── chapters
    │   ├── abstract.tex
    │   ├── appendix.tex
    │   └── chapters.tex
    ├── images
    ├── literature.bib
    ├── thesis-final.tex
    ├── class.cls
    └── acronyms.cls
```

Literatura

[1] Asset definition of Asset at Dictionary.com. Dictionary.com — Meanings a Definitions of Words at Dictionary.com [online]. Copyright © Random House, Inc. 2020 [cit. 10.01.2020]. WWW: https://www.dictionary.com/browse/asset.

[2] Asset Management - LGAM Knowledge Base. Local Government Municipal Knowledge Base - LGAM Knowledge Base [online] [cit. 06.01.2020]. WWW: http://www.lgam.info/asset-management.

[3] 7 Types of Asset Management - Simplicable. [online]. Copyright © 2010 [cit. 06.01.2020]. WWW: https://simplicable.com/new/types-of-asset-management.

[4] What is IT Asset Management? Let IAITAM be your guide.. IAITAM - ITAM Education, Membership a Conferences [online]. Copyright ©INTERNATIONAL ASSOCIATION OF INFORMATION TECHNOLOGY ASSET MANAGERS, INC. ALL RIGHTS RESERVED. [cit. 06.01.2020]. WWW: https://iaitam.org/what-is-it-asset-management.

[5] What is a Patch? - Definition from Techopedia. Techopedia - Where IT a Business Meet [online]. Copyright © 2020 Techopedia Inc. [cit. 29.04.2020]. WWW: https://www.techopedia.com/definition/24537/patch.

[6] What is Patch Management? - Definition from Techopedia. Techopedia - Where IT a Business Meet [online]. Copyright © 2020 Techopedia Inc. [cit. 06.01.2020]. WWW: https://www.techopedia.com/definition/13835/patch-management.

[7] Patch Management — Automate Device Updates — Miradore. Miradore — See Your IT More Clearly [online]. Copyright © Miradore Ltd. [cit. 06.01.2020]. WWW: https://www.miradore.com/patch-management.

[8] Corpus Solutions. Corpus Solutions [online]. Copyright © 2020 Corpus Solutions a.s. [cit. 06.01.2020]. WWW: https://www.corpus.cz/index.html.

[9] Úvod do problematiky informační bezpečnosti. Ing. Jan Bareš, CISA - PDF Stažení zdarma. Představujeme Vám pohodlné a bezplatné nástroje pro publikování a sdílení informací. [online]. Copyright © DocPlayer.cz [cit. 06.01.2020]. WWW: https://docplayer.cz/3791707-Uvod-do-problematiky-informacni-bezpecnosti-ing-jan-bares-cisa.html.

[10] JIRA Software - AppMania - správný software pro Váš business. [online]. Copyright © 2011 [cit. 29.04.2020]. WWW: https://apps.managementmania.com/cs/products/jira-software.

[11] IT Asset Management Software, ITAM, Asset Lifecycle Management - ManageEngine AssetExplorer. ManageEngine - IT Operations a Service Management Software [online]. WWW: https://www.manageengine.com/products/asset-explorer.

[12] [online]. Copyright © 2020 Atlassian [cit. 06.01.2020]. WWW: https://marketplace.atlassian.com/apps/1212137/insight-asset-management?hosting=cloud&tab=overview.

[13] Freshservice ITSM System — ITIL-aligned service desk software [online]. Copyright © Freshworks Inc. All Rights Reserved [cit. 06.01.2020]. WWW: https://freshservice.com.

[14] Asset Tracking, Asset Management Software Apps — Asset Panda. Asset Tracking, Asset Management Software Apps — Asset Panda [online]. Copyright © www.assetpanda.com [cit. 06.01.2020]. WWW: https://www.assetpanda.com.

[15] Enterprise Patch Management — ManageEngine Patch Manager Plus. ManageEngine - IT Operations a Service Management Software [online]. WWW: https://www.manageengine.com/patch-management.

[16] ITarianWindows Patch Management Software— Get it. ITarianIT Management Platform [online]. WWW: https://www.itarian.com/patch-management/windows-patch-management.php.

[17] Patch Management Software — SolarWinds. IT Management Software Monitoring Tools — SolarWinds [online]. Copyright © 2020 SolarWinds Worldwide, LLC. All rights reserved. [cit. 06.01.2020]. WWW: https://www.solarwinds.com/patch-manager.

[18] Patch Management Software - For Windows More — Kaseya VSA. IT Management Software a Monitoring Solutions — Kaseya [online]. Copyright © 2019 Kaseya Limited [cit. 06.01.2020]. WWW: https://www.kaseya.com/products/vsa/patch-management.

[19] GitHub - vuejs/vuex: Centralized State Management for Vue.js.. The world's leading software development platform · GitHub [online]. Copyright © 2020 GitHub, Inc. [cit. 19.03.2020]. WWW: https://github.com/vuejs/vuex.

[20] List of Top JavaScript Frameworks 2020 For Front-End Developers. Learn to code — freeCodeCamp.org [online]. WWW: https://www.freecodecamp.org/news/complete-guide-for-front-end-developers-javascript-frameworks-2019.

[21] Getting Started — BootstrapVue. BootstrapVue [online]. WWW: https://bootstrap-vue.org/docs.

[22] What exactly is Node.js?. Learn to code — freeCodeCamp.org [online]. WWW: https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5.

[23] 10 Node Frameworks to Use in 2019 — Scotch.io. Top Shelf Web Development Training — Scotch.io [online]. Copyright © Scotch.io, LLC [cit. 19.03.2020]. WWW: https://scotch.io/bar-talk/10-node-frameworks-to-use-in-2019.

[24] What is PostgreSQL. PostgreSQL Tutorial - Learn PostgreSQL from Scratch [online]. WWW: https://www.postgresqltutorial.com/what-is-postgresql.

[25] What is Unified Modeling Language (UML)?. Ideal Modeling Diagramming Tool for Agile Team Collaboration [online]. WWW: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml.

[26] Documentation for Visual Studio Code. Visual Studio Code - Code Editing. Redefined [online]. Copyright © 2020 [cit. 13.04.2020]. WWW: https://code.visualstudio.com/docs.

[27] pgAdmin 4 — pgAdmin 4 4.20 documentation. pgAdmin - PostgreSQL Tools [online]. WWW: `https://www.pgadmin.org/docs/pgadmin4/4.11/index.html`.

[28] How to Secure Your REST API using Proven Best Practices — Stormpath. Identity User Management API — Stormpath [online]. WWW: `https://stormpath.com/blog/secure-your-rest-api-right-way`.

[29] A Beginners Guide To Cron Jobs - OSTechNix. OSTechNix - Open Source - Technology - Linux And Unix [online]. Copyright © 2020. All Rights Reserved. WWW: `https://www.ostechnix.com/a-beginners-guide-to-cron-jobs`.

[30] What is Functional Testing? Types Examples (Complete Tutorial). Meet Guru99 - Free Training Tutorials Video for IT Courses [online]. WWW: `https://www.guru99.com/functional-testing.html`.