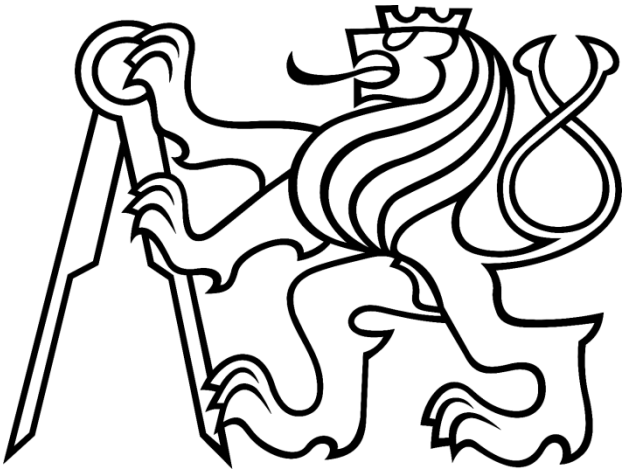
České vysoké učení technické v Praze

Fakulta elektrotechnická



Diplomová práce

Registr členů klubu Hlávkova kolej

Bc. Martin Dendis

Vedoucí práce: Ing. Jan Kubr

Studijní program: Otevřená informatika, Magisterský

Obor: Softwarové inženýrství

9. ledna 2018

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Dendis Martin

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: Registr členů klubu Hlávkova kolej

Pokyny pro vypracování:

Proveďte analýzu stávajícího řešení Registru členů klubu Hlávkova kolej. Proveďte analýzu podobných řešení stejné problematiky.
Navrhněte a implementujte funkcionalitu Registru členů Klubu Hlávkova kolej. Aplikaci navrhněte jako webovou nad technologií PHP/JavaScript.
Aplikace umožní:

- evidovat členy klubu a spravovat jejich osobní informace;
- omezit výčet funkcí aplikace na základě uživatelských rolí;
- řídit přístup členů k poskytovaným službám klubu;
- přístup členům k vlastním osobním informacím a nastavení služeb;
- periodické stahování transakcí z bankovního účtu;
- párování transakcí s účty členů;
- generování reportů.

Specificky se zaměřte na tyto oblasti:

- responzivní design aplikace;
- modulární architekturu aplikace;
- propojení s OpenLDAP databází;
- automatizování procesů.

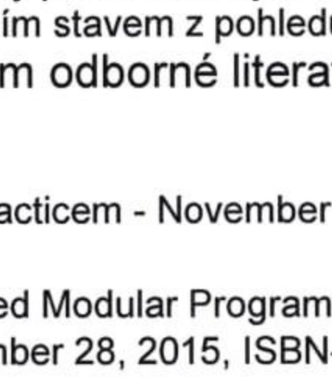
Navrhněte vhodné funkční a zátěžové testy. Testy proveďte a vyhodnoťte.
Zhodnoťte přínos aplikace ve srovnání s původním stavem z pohledu uživatele a správce systému.

Seznam odborné literatury:

[1] Matt Zandstra - PHP Objects, Patterns, and Practicem - November 26, 2013 - ISBN-10: 1430260319, ISBN-13: 978-1430260318.
[2] Steve Prettyman - Learn PHP 7: Object Oriented Modular Programming using HTML5, CSS3, JavaScript, XML, JSON, and MySQL (1st ed. Edition) December 28, 2015, ISBN-10: 1484217292, ISBN-13: 978-1484217290.
[3] Baron Schwartz, Peter Zaitsev, Vadim Tkachenko ? High Performance MySQL: Optimization, Backups, and Replication March 30, 2012 - ISBN-10: 1449314287, ISBN-13: 978-1449314286.
[4] Kévin Dunglas - Persistence in PHP with Doctrine ORM December 18, 2013 - ISBN-10: 1782164103, ISBN-13: 978-1782164104.
[5] Matt Butche - Mastering OpenLDAP: Configuring, Securing and Integrating Directory Services August 31, 2007 - ISBN-10: 1847191029, ISBN-13: 978-1847191021

Vedoucí: Ing. Jan Kubr

Platnost zadání do konce letního semestru 2017/2018



prof. Dr. Michal Pěchouček, MSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 27.2.2017

Poděkování

Rád bych zde poděkoval panu Ing. Janu Kubrovi za jeho rady a čas, které mi věnoval při řešení dané problematiky. Dále bych rád poděkoval svým rodičům za jejich velkou podporu během celého studia. V neposlední řadě bych pak chtěl poděkovat všem správcům sítě z ostatních kolejí ČVUT za jejich ochotu a čas, který mi věnovali při provádění rešerše existujících informačních systémů na ostatních kolejích ČVUT.

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 dne ……. ……

Podpis autora práce

Abstrakt

Cílem této práce je navrhnout a implementovat nový informační systém pro evidenci členů klubu Hlávkova kolej a řízení přístupu ke službám klubu, který nahradí stávající nevhodné řešení. Práce popisuje celý vývojový cyklus od analýzy stávajícího řešení a jeho nedostatků, která pak posloužila jako podklad pro sumarizaci požadavků pro systém nový, analýzy existujících podobných řešení s ohledem na jejich využití, po návrh, implementaci, testování a nasazení nového systému do produkčního prostředí. V závěru této práce je pak provedeno zhodnocení přínosu nového systému a představeny návrhy možného dalšího rozšíření.

Abstract

The aim of this work is to design and implement a new information system for registration of members of club Hlávkova kolej and for access control of club services, which will replace the current inappropriate solution. The thesis describes the entire development cycle from the analysis of the current solution and its shortcomings, which served as a basis for summarizing the requirements for the new system, analysis of existing similar solutions with regard to their use, designing, implementing, testing and deploying the new system into the production environment. At the end of this work, an evaluation of the contribution of the new system is presented along with proposals for its further development.

Obsah

1. Úvod

Okolo roku 1997 založila skupina studentů na Strahovských kolejích experimentální počítačovou síť, která v polovině roku 1998 přerostla rozměry soukromé lokální sítě. Síť proto řešila řadu legislativních i finančních problémů a jako vhodné řešení se ukázalo začlenit fungování sítě do nově vzniklé Studentské unie ČVUT v podobě autonomního klubu. Jelikož se tento model osvědčil, začaly v rychlém tempu vznikat další kolejní kluby, které se postupem času začlenily do Studentské unie ČVUT. Časem se pak seznam nabízených služeb rozrůstal stejně tak jako seznam aktivních členů[1] a dnes již kluby na většině kolejí nabízejí svým členům i další služby, jako například hostování školních projektů, provoz fitcenter, organizaci volnočasových aktivit a mnoho dalšího, přičemž hlavním posláním všech kolejních klubů stále zůstává podpora studia, a tedy i zajištění kvalitního připojení k vysokorychlostnímu internetu a přístupu k online studijním materiálům studentům ubytovaným na kolejích ČVUT [1].

Pro účely evidence členů klubu, správu jejich osobních údajů, řízení přístupu k jednotlivým službám a přehledu nad uhrazenými členskými příspěvky vznikly různé informační systémy, přičemž každý kolejní klub si zpravidla vyvíjí vlastní informační systém v závislosti na svých konkrétních potřebách, nastavené politice a výčtu nabízených služeb.

Klub Hlávkova kolej (dále jen jako klub HK) pro tyto účely používá od začátku roku 2011 informační systém RUPS (Registr Uživatelů Počítačové Sítě), který v rámci své bakalářské práce navrhla a implementovala Bc. Adéla Dragounová [2].

Před nasazením prvního informačním systémem RUPS se od roku 2010, kdy klub HK vznikl, vedla evidence členů pouze v Excel formátu a veškerý síťový hardware a služby se tak musely konfigurovat manuálně, což obzvláště na přelomu semestrů vedlo k velkému časovému vytížení registrátorů a administrátorů počítačové sítě.

Zavedení informačního systému RUPS tak vedlo ke zpřehlednění a zjednodušení správy členů klubu HK, odpadla potřeba konfigurovat síťové prvky manuálně, díky čemuž se většina procesů spojených se správou zrychlila a zefektivnila.

1.1 Motivace

Informační systém RUPS bohužel není zcela intuitivní, některé jednoduché úlohy vyžadují vykonání až zbytečně mnoha úkonů a u některých úloh je navíc potřeba přesně dodržet předepsaný postup, jelikož v opačném případě hrozí, že dojde k chybnému nastavení síťových prvků. Oprava konfigurace pak musí být provedena manuálně přímo na konkrétním prvku.

Systém navíc nebyl od doby svého vzniku dále aktivně vyvíjen a nedostačuje tak dnes již běžným standardům, nepodporuje responzivní design pro mobilní zařízení a problémová je rovněž i kompatibilita s některými novými verzemi prohlížečů.

Drtivá většina kódu aplikace je napsána procedurálně a OOP[2] (když už je použito) se používá hlavně pro implementaci DAO[3] funkcionality. Pro přístup k databázi je pak použito nativních PHP[4] funkcí, což vede k potenciálně možným bezpečnostním rizikům, například z důvodu opomenutí sanace vstupních dat.

Od roku 2011 také došlo k drobným změnám konfigurace síťových prvků a služeb a ke změně bankovního spojení, v důsledku čehož přestaly fungovat některé automatizované skripty, které generují pro tyto služby konfigurace a periodicky stahují informace o nových platbách.

---

[1] Aktivní člen – člen klubu, který se aktivně podílí na dění v klubu (většinou člen představenstva, nebo správce nějaké služby).

[2] OOP – Object Oriented Programming, programovací paradigma.

[3] DAO – Data Access Object, objekt poskytující rozhraní pro přístup do databáze.

[4] PHP – PHP: Hypertext Pre-processor, skriptovací programovací jazyk.

Jako největší problém lze ale jednoznačně označit stávající řešení infrastruktury počítačové sítě, která nyní umožňuje souběžné připojení maximálně 242 koncových zařízení, tudíž může mít každý člen klubu připojeno do sítě pouze jedno zařízení. Toto omezení je tak v době rozmachu chytrých telefonů a tabletů zásadním nedostatkem.

Na začátku roku 2016 proto došlo k rozhodnutí renovovat celou infrastrukturu počítačové sítě, pokrýt celou kolej Wi-Fi signálem a navrhnout a implementovat nový informační systém, který odstraní nedostatky stávajícího řešení a zároveň zjednoduší a maximálně automatizuje veškeré prováděné úkony tak, aby byl nový systém v rámci mezí plně soběstačný.

1.2 Cíle práce

Cílem této práce je vytvořit nový informační systém pro klub HK, který odstraní nedostatky stávajícího řešení a maximálně zjednoduší evidenci klubových členů a správu jejich služeb.

Proto bude nejprve provedena analýza stávajícího informačního systém klubu HK a jeho nedostatků. Na základě zjištěných informací budou následně sepsány ucelené požadavky na nový informační systém klubu HK, provedena analýza podobných existujících řešení s ohledem na jejich využití, navrhnut nový informační systém a následně bude tento systém implementován a otestována jeho funkčnost. Na závěr bude provedeno zhodnocení celého systému, zdali odpovídá novým požadavkům a představeny návrhy na jeho další možný rozvoj.

Jelikož se některé požadavky na systém přímo opírají o požadavky počítačové sítě, bude do této práce rovněž zahrnuta zjednodušená analýza původní a nové infrastruktury počítačové sítě se sumarizací jejich požadavků.

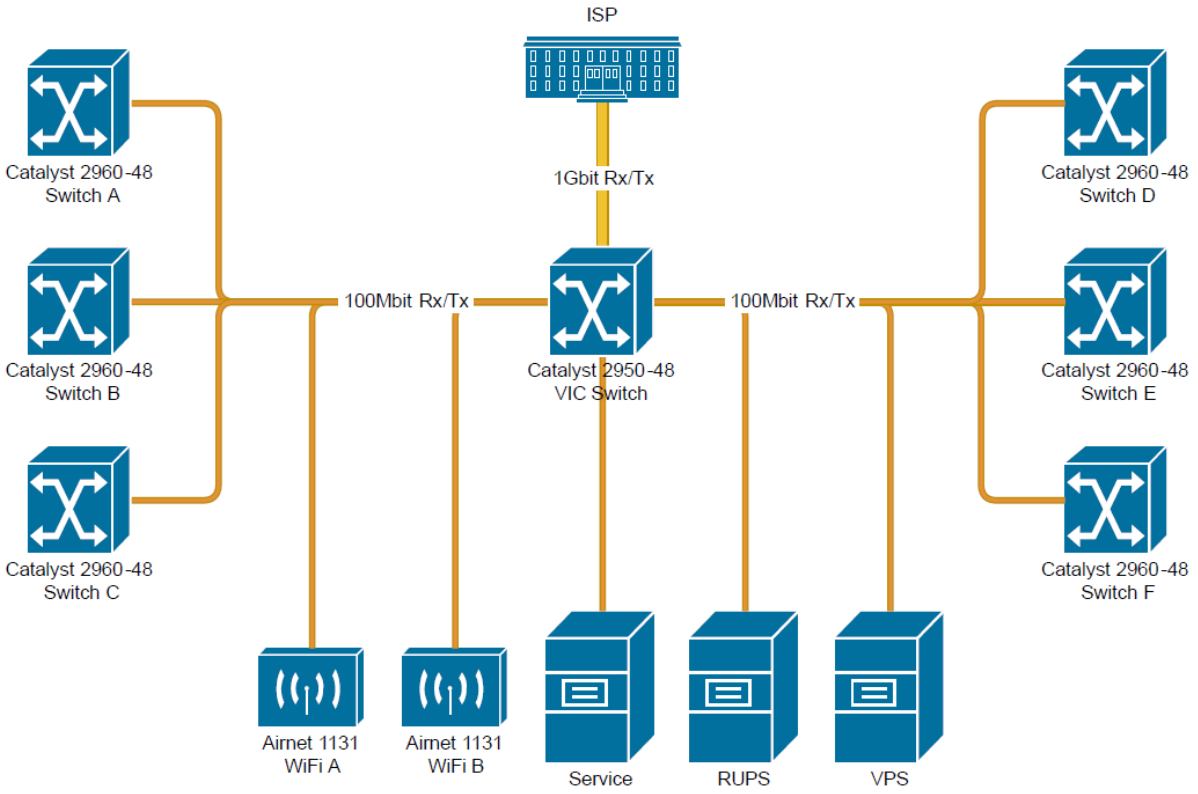
2. Analýza stávajícího řešení

Pro lepší pochopení stávajícího řešení včetně jeho požadavků a nedostatků bude nejprve provedena krátká analýza síťového řešení, pro které bylo navrhnuto. Následně bude provedena analýza stávajícího informačního systému RUPS a jeho nedostatků, která poslouží jako podklad pro požadavky nového systému.

2.1 Analýza počítačové sítě

2.1.1 Topologie sítě

Zjednodušené schéma zapojení síťových prvků znázorňuje přiložený Obrázek 1.



Obrázek 1 – Zjednodušená topologie stávající infrastruktury počítačové sítě

Na Hlávkovu kolej je optikou přivedena 1Gbit konektivita, která je vyvedena do switche od VIC ČVUT. Na tento prvek jsou pak přímo napojeny veškeré servery, Wi-Fi AP[5] *A* až *B* a patrové switche *A* až *F*, na něž jsou pak přímo připojeni klienti[6].

Každý síťový prvek, server a klient má přiřazenou veřejnou IP z rozsahu 147.32.98.0/24, která je na síťových prvcích a serverech nastavena staticky a klientům je přiřazována DHCP[7] serverem dynamicky na základě předem definovaných rezervací podle MAC adresy klienta.

Patrové switche *A* až *F* mají na přístupových portech (kam se připojují klienti) nastavenou port-security tak, že se na něj mohou připojit pouze klienti s MAC adresou, která je na tomto portu registrována v informačním systému RUPS.

---

[5] AP – Access Point, bezdrátový přístupový bod do lokální sítě.
[6] Klient – zařízení, nebo aplikace připojující se k nějaké službě.
[7] DHCP – Dynamic Host Configuration Protocol, služba pro automatickou konfiguraci klientů připojených do počítačové sítě.

Obdobné nastavení je pak aplikováno pro bezdrátovou síť s tím rozdílem, že klienti nejsou vázáni na konkrétní AP a jsou jim přiřazovány IP adresy z privátní sítě 192.168.1.0/24. Jako router pro tuto privátní síť pak slouží servisní stroj (*Service*).

2.1.2 Síťové služby a jejich požadavky na systém

Na servisním stroji (*Service*) běží síťové služby DNS[8], DHCP, MX[9] a také NAT[10] pro bezdrátovou síť. Níže si tyto služby představíme blíže.

2.1.2.1 DNS server

DNS server je autoritativním serverem pro doménu *hk.cvut.cz***.** Pro konfiguraci DNS záznamů slouží textová šablona v systému RUSP, do které systém doplňuje překlady pro veřejné IP adresy klientů. Pro doménové jméno klienta je pak použita složenina z UID[11] a písmena, které jednoznačně reprezentuje uživatelovo zařízení (značí se od písmena *A* dále).

Systém RUPS generuje novou DNS konfiguraci každou hodinu. Po vygenerování nové konfigurace je soubor nahrán na servisní stroj a proveden restart DNS serveru.

Systém RUPS neumožňuje upravovat DNS záznamy přes grafické rozhraní a úpravy se tak musí dělat manuálně přímo v šabloně pro konfiguraci.

Během analýzy bylo navíc zjištěno, že DNS server je chybně nakonfigurován, v důsledku čehož neodpovídá na žádné požadavky na překlad. V praxi to pak znamená, že se změny v DNS záznamech projeví až po jejich propagaci na DNS servery ČVUT, což nějakou dobu trvá.

2.1.2.2 DHCP server

DHCP server odpovídá na konfigurační požadavky pro drátovou síť 147.32.98.0/24 a bezdrátovou síť 192.168.1.0/24.

Stejně jako u DNS serveru je i pro DHCP server konfigurace generována za použití textové šablony, do které systém RUPS doplňuje IP rezervace pro jednotlivé MAC adresy klientů, a která je následně přenesena na servisní stroj pro aplikování změn.

Nová DHCP konfigurace je generována každých 15 minut.

2.1.2.3 MX server

Každý člen klubu má klubový emailový alias ve tvaru *uživatelské_jméno@hk.cvut.cz*, který je nasměrován na emailovou adresu, jež člen uvedl při registraci.

MX server je nastaven tak, aby pouze přesměrovával příchozí poštu z email aliasů na registrované emaily členů, a neumožňuje tak vytváření emailových schránek ani přihlášení přes protokoly IMAP, POP3 a SMTP.

Z tohoto důvodu musejí být veškeré emaily, které informační systém RUPS zasílá, posílány přímo ze stroje, na kterém systém RUPS běží, v důsledku čehož drtivá většina těchto emailů končí ve SPAMu kvůli jejich nedůvěryhodnosti.

Generování nového nastavení pro MX server probíhá každou hodinu.

2.1.3 Síťové prvky a jejich požadavky na systém

Část síťových požadavků v podobě generování konfiguračních souborů již byla vyjmenována v popisu jednotlivých síťových služeb. Zbylé požadavky se tak týkají přímo konfigurace síťových prvků.

---

[8] DNS – Domain Name System, služba pro překlad doménových jmen na IP adresy.
[9] MX – Mail Exchange, služba pro výměnu emailových zpráv.
[10] NAT – Network Address Translation, služba pro překlad zdrojových a cílových IP adres.
[11] UID – Uživatelské identifikační číslo.

2.1.3.1 Konfigurace patrových switchů

Jak již bylo dříve uvedeno, každý patrový switch má na přístupových portech nastavenou port-security, v důsledku čehož se na daný port může připojit pouze klient s MAC adresou registrovanou v systému RUPS.

Konfigurace je na switche nahrávána pomocí skriptu v jazyce Expect každých 15 minut. Celý proces probíhá tak, že se skript nejprve připojí a autentifikuje na switchi a následně na něm spustí sadu předem připravených příkazů, které nakonfigurují jednotlivé porty. Příkazy pro změnu konfigurace portu generuje automaticky systém RUPS při úpravě nastavení uživatelova zařízení. Po aktualizaci konfigurace switche je obsah souboru s příkazy smazán.

Díky tomuto přístupu se na switch aplikují pouze příkazy pro provedení změn a nenahrává se tak celá konfigurace switche, což by mělo za důsledek dočasné odpojení všech připojených zařízení. Zároveň to ale znamená, že v případě chybně vygenerovaného nebo neúplného seznamu příkazů může dojít k zablokování některých portů na switchi (což se může při nedodržení některých postupů při editaci zařízení v systému RUPS doopravdy stát).

2.1.3.2 Konfigurace Wi-Fi AP

Obdobné omezení, jako u patrových switchů, platí i pro Wi-Fi AP, na které se mohou přihlásit pouze zařízení, která mají registrovanou MAC adresu v systému RUPS.

Pro tyto účely systém RUPS generuje každých 15 minut seznam MAC adres, které se mohou k AP připojit a následně je tento seznam nahrán na jednotlivá Wi-Fi AP.

2.1.3.3 NAT

Kvůli omezenému počtu veřejných IP adres byla pro potřeby Wi-Fi vytvořena privátní síť, pro kterou dělá servisní stroj NAT. V praxi to pak znamená, že veškerá zařízení připojená do této sítě jsou schována za IP adresou servisního stroje, a protože na síti nebylo nastaveno logování NATu, stala se Wi-Fi velmi rychle oblíbenou pro stahování autorsky chráněného obsahu přes peer-to-peer sítě.

Z tohoto důvodu byla Wi-Fi po cca dvou letech svého provozu dočasně odstavena do doby, než se tento problém podaří uspokojivě vyřešit.

2.1.4 Nevýhody tohoto řešení

Současné řešení sítě má kvůli jeho příliš velké jednoduchosti spoustu omezení a nevýhod, které by mělo nové řešení odstranit. Mezi ty nejzávažnější patří následující:

- Veškeré síťové prvky mají přiřazenou veřejnou IP adresu, což nejen že otevírá dveře potenciálním útočníkům, ale zároveň tak není hospodárně nakládáno s veřejným IP blokem.
- V důsledku malého a neekonomicky využívaného bloku veřejných IP adres je možné teoreticky do sítě připojit jen cca 240 zařízení členů klubu, což je vzhledem ke kapacitě Hlávkovy koleje (235 lůžek) žalostně málo.
- Uživatelé sice mají přiřazenou veřejnou IP adresu, díky čemuž lze dohledat viníka v případě porušení předpisové základny klubu, nicméně tuto IP adresu si může každý člen změnit ručně. Systém nijak nehlídá, zdali klient používá IP adresu, kterou má přidělenu v systému RUPS.
- V současné době síť neumožňuje spolehlivý (dohledatelný) provoz Wi-Fi sítě.
- Zasílané emaily jsou nedůvěryhodné a padají do SPAMu.
- Chybně vygenerovaná konfigurace pro DHCP, switch nebo AP může ochromit část nebo celou síť a oprava se pak musí provést manuálně.
- Výpadek nebo odstávka servisního stroje ochromí celou počítačovou síť.
- Jelikož je všem klientům přiřazena veřejná IP adresa a na síti není předřazen žádný firewall, mohou se spatně zabezpečená zařízení členů stát snadným terčem různých útoků.

2.1.5 Zhodnocení analýzy stávající sítě

Analýzou bylo zjištěno, že jediné požadavky na informační systém jsou generování konfiguračních souborů pro DNS, DHCP a MX servery a dále pak generování konfigurace pro patrové switche a Wi-Fi AP, ke kterým dochází v 15 minutových až hodinových intervalech.

Celý koncept počítačové sítě byl navržen tak, aby její správa byla co nejjednodušší a aby bylo možné v případě výpadku některého síťového prvku tento prvek jednoduše a rychle nahradit i za cenu toho, že některé síťové služby budou dočasně nedostupné, nebo dojde k vyřazení některých ochranných mechanizmů (port-security).

Uživatelským zařízením jsou na metalické síti přidělovány IP adresy z veřejného bloku, díky čemuž lze v případě porušení pravidel o využívání počítačové sítě nebo při činnosti, která je v rozporu se zákonem ČR, dohledat inkriminovaný stroj a jeho majitele. Tato funkce ale nefunguje, pokud je uživatel připojen přes bezdrátovou síť Wi-Fi, protože pak do internetu přistupuje pod IP adresou servisního stroje.

Kvůli jednoduchosti stávajícího řešení má počítačová síť řadu závažných nedostatků, a proto bude nad rámec této diplomové práce navrhnuta a nakonfigurována nová počítačová sít, viz kapitola 3.

2.2 Analýza informačního systému RUPS

Nejprve bude proveden rozbor uživatelských rolí, uveden jejich význam v systému a představen jejich soupis pravomocí. Následně bude provedena analýza dostupné funkcionality a sepsány zjištěné nedostatky a objevené chyby. Rovněž bude provedena i analýza implementace stávajícího systému za účelem jejího potenciálního využití při návrhu nového informačního systému.

V závěru této kapitoly pak bude provedeno zhodnocení získaných poznatků, které zároveň poslouží jako podklad pro požadavky na nový informační systém.

Více informací o informačním systému RUPS může být nalezeno v bakalářské práci Adély Dragounové [2]. Postupem času byl ale systém lehce upraven a níže uvedená analýza a názvosloví (například uživatelských rolí) se tak nemusí přesně shodovat s původním návrhem popsaným v bakalářské práci Adély Dragounové.

2.2.1 Uživatelské role

Informační systém RUPS implementuje celkem 6 uživatelských rolí, které částečně opisují hierarchickou strukturu členů v klubu HK. Níže si tyto role detailněji představíme.

2.2.1.1 Nepřihlášený uživatel

Tento uživatel nemá v systému téměř žádné pravomoci a jediné, co může, je se do systému přihlásit nebo požádat o reset hesla.

2.2.1.2 Aktivní uživatel

Dědí od *Nepřihlášeného uživatele*. Po přihlášení může:

- zkontrolovat veškeré informace, které jsou o něm v informačním systému drženy;
- zkontrolovat své platby členských příspěvků;
- provést změnu hesla a emailu;
- prostřednictvím informačního systému zaslat registrátorovi nebo administrátorovi zprávu.

Editace jiných osobních údajů není povolena.

2.2.1.3 Neaktivní uživatel

Má stejné pravomoci jako *Aktivní uživatel* a reprezentuje bývalého člena klubu HK nebo člena, který včas neuhradil členský příspěvek.

Tento člen má odepřen přístup do počítačové sítě a k dalším službám klubu HK.

2.2.1.4 Zabanovaný uživatel

Má stejné pravomoci jako *Neaktivní uživatel* a reprezentuje člena klubu HK, kterému bylo pozastaveno členství z důvodu hrubého porušení předpisové základny klubu HK nebo nadřazených předpisů.

2.2.1.5 Registrátor

Má stejné pravomoci jako *Aktivní uživatel*, a navíc má pravomoci pro:

- registraci a editaci členů klubu HK včetně jejich síťových zařízení;
- nastavení uživatelovy role na *Aktivní uživatel*, *Neaktivní uživatel* a *Zabanovaný uživatel*;
- udělení banu včetně jeho odůvodnění a nastavení délky trvání;
- rozesílání hromadných zpráv členům;
- zobrazení logu provedených akcí v systému;
- zobrazení seznamu volných IP;
- přijímání zpráv od členů.

2.2.1.6 Administrátor

Má stejné oprávnění jako *Registrátor* rozšířené o možnost:

- nastavit uživatelům role *Registrátor* a *Administrátor*;
- zobrazit a přiřadit ke členům nepřiřazené příchozí platby (které se jinak automaticky párují podle variabilního symbolu);
- udělovat odpustky plateb členských příspěvků;
- nastavit čas spouštění skriptů, které generují konfigurační soubory pro síťové služby a prvky, viz kapitoly 2.1.2 a 2.1.3.

2.2.2 Dostupná funkcionalita

Výpis funkcionality systému byl již částečně představen při analýze pravomocí jednotlivých uživatelských rolí. Nyní detailněji podrobíme jednotlivé funkce analýze a určíme jejih použitelnost a případné nedostatky.

2.2.2.1 Správa hesla

Systém generuje heslo každému novému členovi a není tak možné jej při registraci člena nastavit. Po prvním přihlášení člena do systému si proto většina uživatelů jako první věc nastaví heslo nové.

Pro reset hesla je potřeba znát své UID (uživatelské ID), což se bohužel postupem času ukázalo jako nepraktické řešení, jelikož většina členů klubu si své UID nepamatuje. Řešením by tak mohlo být vyžadovat pro reset hesla například uživatelské jméno, nebo ještě lépe registrovanou emailovou adresu.

Dalším problémem pak je, že systém nevyžaduje žádnou autorizaci požadavku na změnu hesla a rovnou uživateli vygeneruje a zašle nové heslo. Toto chování je také nevyhovující, jelikož nový návrh počítačové sítě bude kvůli zvýšené bezpečnosti ověřovat uživatele i podle uživatelského jména a hesla (viz kapitola 3). Reset hesla by tak v praxi měl za následek odpojení uživatele od sítě, a tedy i od internetu, v důsledku čehož by si nemohl zjistit heslo nové.

Z těchto důvodů nebude prozatím do nového systému uvažována implementace resetu hesla a uživatel tak bude muset požádat registrátora nebo administrátora o nastavení hesla nového.

2.2.2.2 Registrace a editace členů

Při registraci nebo editaci člena vyplní registrátor nebo administrátor *jméno, příjmení, datum narození, adresu, email, číslo pokoje* a nastaví členovi *uživatelské jméno*. Po uložení karty člena je registrátorovi zobrazena hláška *„Uživatel s UID XXXX byl úspěšně vytvořen.“*, a to i při editaci již existujícího uživatele.

Při registraci navíc systém na pozadí automaticky doplní k profilu uživatele i datum registrace a přiřadí mu roli *Aktivní uživatel*.

Zjištěné nedostatky:

- Chybná hláška při ukládání existujícího uživatele.
- Při registraci nového člena formulář nepodporuje znaky s diakritikou a do textových polí pro jméno a příjmení nelze zapsat mezeru. Zajímavostí přitom je, že při editaci již existujícího člena se tento problém již nevyskytuje.

- Pro adresu je systému vyhrazeno pouze jedno textové pole, které tak zároveň obsahuje ulici, č.p., město, PSČ a stát. V případě cizinců tak musí být dodržován předepsaný postup, jinak nelze spolehlivě určit, co z adresy je město, co ulice atd.
- Podle nových stanov SU ČVUT se navíc musí u každého člena evidovat i místo narození. Toto textové pole ve stávajícím systému chybí.
- K profilu člena nelze vést další přidružené informace, např. ve formě poznámky.
- Při registraci nelze členovi nastavit prvotní heslo.

2.2.2.3 Evidence banů

Každému členovi může být z důvodu nerespektování a hrubého porušení předpisové základny klubu HK nebo nadřazených předpisů udělen trest v podobě pozastavení členství v klubu, a tedy i odepření členských výhod. Při opakovaném porušování těchto předpisů se navíc může délka trestu zvyšovat.

K tomuto účelu je na kartě člena vedena i evidence banů (trestů) spolu s jejich odůvodněním a délkou trvání, které spravuje registrátor a administrátor informačního systému.

Zjištěné nedostatky:

- Po zadání trestu do systému nelze tento trest dále editovat nebo smazat a v případě potřeby je tak nutno provést změny manuálně přímo v databázi.
- Po zadání trestu systém automaticky nepozastaví členovi členství v klubu ani jej neodpojí od počítačové sítě. Registrátor tak musí navíc změnit uživateli roli na *Zabanovaný uživatel* a všem registrovaným síťovým zařízením uživatele odebrat IP a port patrového switche, ke kterému se připojují.
- Po vypršení trestu systém automaticky neaktivuje členství uživatele a musí se tak dělat opět manuálně.

2.2.2.4 Evidence členských příspěvků

Na kartě člena lze rovněž vidět přehled jeho uhrazených členských příspěvků. Členské příspěvky se na denní bázi získávají přes API banky a jsou automaticky párovány podle variabilního symbolu, kterým je UID uživatele.

Pro případy, kdy člen při úhradě příspěvku zadá chybný variabilní symbol nebo jej nezadá vůbec, má administrátor systému k dispozici rozhraní, přes které může ručně přiřadit platbu ke kartě uživatele po předložení výpisu z účtu nebo jiného dokumentu, který potvrzuje provedenou platbu.

V případě, kdy člen pro klub vykoná během předešlého zúčtovacího období nějakou záslužnou činnost, může představenstvo klubu rozhodnout o prominutí platby za následující zúčtovací období.

Zjištěné nedostatky:

- Pokud člen zadá ve variabilním symbolu platné UID jiného člena nebo pokud administrátor nepřiřazenou platbu přiřadí ke špatnému uživateli, nelze toto propojení přes rozhraní systému zrušit a musí tak být provedeno manuálně přímo v databázi.
- Stejné omezení pak platí i při chybném nebo neoprávněném udělení odpustku za platbu.
- K odpustkům nelze přidat poznámku s odůvodněním, proč byl členovi odpustek udělen.
- Informační systém získává z banky pouze část dostupných informací o platbách a v některých případech tak administrátor nemá dostatek informací pro identifikaci a přirazení platby.
- Po přiřazení platby ke členovi již nelze zobrazit doplňující informace o platbě.
- Po změně banky (a tedy i API) přestaly fungovat automatizované skripty a informace o platbách se tak musí do systému zadávat manuálně.
- Systém neumí automaticky deaktivovat, nebo reaktivovat členství v závislosti na nastavené platební politice.

2.2.2.5 Vyhledávání členů

Registrátorům a administrátorům je k dispozici tabulka s výčtem aktivních členů, ve které jsou k dispozici sloupce obsahující *UID, jméno, příjmení, adresu, datum narození* a *číslo pokoje*, na kterém je člen ubytován. Po rozkliknutí řádku se pak načte a zobrazí karta člena.

Stejná tabulka je také zobrazena při vyhledávání člena, při kterém je možné vyhledávat na základě *UID, jména, příjmení* a *data narození*.

Zjištěné nedostatky:

- Vyhledávání nad členy je citlivé na velikost znaků a diakritiku, což je obzvláště u některých cizojazyčných jmen problém, a navíc je toto omezení nepříjemné.
- Nelze vyhledávat nad IP adresou a při dohledávání člena z důvodu nekalé aktivity na síti je tak potřeba vyhledávat přímo v databázi informačního systému.
- Tabulka s výpisem uživatelů neobsahuje informaci o přidělených IP adresách a stavu členství.

Časem se ukázalo, že vyhledávání členů je natolik neintuitivní, že většina registrátorů a administrátorů vyhledává členy přímo nad tabulkou aktivních členů za pomoci zabudovaných vyhledávacích funkcí v internetovém prohlížeči, nebo manuálně v MySQL databázi.

2.2.2.6 Evidence síťových zařízení

U každého člena je držena evidence všech jeho zařízení, se kterými se připojuje do počítačové sítě klubu. U síťového zařízení je povinně evidována ethernetová MAC adresa a k ní přidělená ethernetová IP adresa a port patrového switche, ke kterému lze zařízení připojit. Navíc lze u každého zařízení nepovinně evidovat i jeho Wi-Fi MAC adresu a k ní přiřazenou IP adresu.

Zjištěné nedostatky:

- Stejně jako u banů a odpustků nelze jednou přidané zařízení ze systému smazat a musí tak být provedeno manuálně přímo v databázi systému.
- Nelze registrovat čistě Wi-Fi zařízení (mobil, tablet), jelikož systém vyžaduje vyplnění ethernetové MAC adresy.
- Systém umožňuje zadat MAC adresu v různých formátech, ale ve skutečnosti je možné adresu zadat pouze ve formátu s pomlčkami (tj. *XX-XX-XX…*). Důvodem je to, že systém neprovádí normalizaci formátu MAC adresy a do generovaných konfiguračních souborů pak propisuje adresu přesně tak, jak byla zadána registrátorem, v důsledku čehož pak může přestat fungovat DHCP server a některé další služby.
- Při změně portu switche nebo MAC adresy zařízení musí být nejprve odebrán port switche, následně musí být konfigurace zařízení uložena a až pak lze nastavit nový port a MAC adresu. Při nedodržení tohoto postupu dojde k vygenerování neúplného aktualizačního skriptu pro patrový switch, což může mít za následek zablokování portu na switchi.
- IP adresy musí registrátor k zařízení manuálně přiřazovat z tabulky volných IP adres, což je nepraktické.
- Při zadávání IP adresy se nekontroluje její správnost, tj. že ethernet IP adresa bude z rozsahu 147.32.98.0/24a Wi-Fi IP adresa z rozsahu 192.168.1.0/24.

2.2.2.7 Systémový log

Systémový log slouží primárně pro hlášení chyb, které nastaly v informačním systému společně s datem a časem, kdy k nim došlo a jaký uživatel akci prováděl. Navíc hlásí některé aktivity uživatelů v systému, jako je přihlášení a odhlášení uživatele a že byl některý uživatel upraven.

Zjištěné nedostatky:

- Systém sice loguje chyby aplikace, nicméně tyto chyby nijak nehlásí administrátorovi a ten tak musí čas od času kontrolovat log manuálně, kvůli čemuž nemusí zareagovat včas, pokud dojde k nějaké závažnější chybě.
- Systém sice loguje změny profilu členů, spolu s informací kdy, kdo, kterého člena upravoval, ale neumí zobrazit informace o tom, jaké změny byly provedeny.

Z praktického hlediska je tak celé stávající logování systémových událostí nepoužitelné.

2.2.2.8 Support systém

Systém umožňuje členům poslat zprávu registrátorovi nebo administrátorovi systému. U zprávy je pak potřeba zadat předmět, text zprávy a vybrat jednoho či více příjemců.

Registrátor, případně administrátor následně po přečtení a zpracování zprávy označí zprávu jako vyřízenou.

Zjištěné nedostatky:

- Příjemce není o přijetí zprávy nijak informován (například emailem) a musí tak pravidelně kontrolovat, zda v systému nemá nové příchozí zprávy.
- Po vyřešení požadavku od člena registrátor označí zprávu jako vyřízenou. Člen ale o vyřízení požadavku není nijak informován.
- Není k dispozici rozhraní, pomocí kterého by si mohl člen zkontrolovat zprávy, které odeslal a zdali byly vyřízeny.
- Není k dispozici rozhraní, pomocí kterého by mohl registrátor jakkoliv uvědomit člena o výsledku jeho dotazu / požadavku, a nemůže se tak ani doptat na případné další potřebné údaje.

Z těchto důvodů je implementovaný support systém prakticky nepoužitelný a veškerá komunikace tak probíhá buď přímo osobně, nebo prostřednictvím emailu.

Jelikož je komunikace přes emaily pohodlnější pro obě strany (členy i registrátory), nebude support systém do nového informačního systému prozatím uvažován.

2.2.2.9 Nastavení spouštění skriptů

Jak již bylo dříve zjištěno, systém generuje v určitých intervalech řadu konfiguračních souborů. Pro tento účel obsahuje systém pro každou generovanou konfiguraci její šablonu a PHP skript, který ji generuje. Skript je pak v předem určených intervalech spouštěn CRONem[12]. Interval spouštění lze nastavit přímo v informačním systému RUPS.

Zjištěné nedostatky:

- Systém sice umožňuje nastavovat interval spouštění jednotlivých skriptů, nicméně z bezpečnostních důvodů (alespoň takto je to odůvodněno v bakalářské práci Adély Dragounové) nelze přes uživatelské rozhraní registrovat skripty nové nebo je pozastavovat či rušit. Tyto úpravy tak musí být prováděny přímo v databázi systému.
- V rozhraní systému nelze manuálně vynutit předčasné spuštění skriptu a nelze tak v případě potřeby urychlit vygenerování a nasazení nové konfigurace.

Vzhledem k těmto omezením a faktu, že přímá úprava CRON konfigurace je mnohem pohodlnější než úprava přes databázi systému, není tato funkcionalita využívána.

2.2.2.10 Odesílání emailů

Uživatelské rozhraní RUPSu umožňuje administrátorovi zaslat hromadný email všem členům, kteří mají v daný okamžik aktivní členství v klubu HK.

Dále systém automaticky generuje a zasílá emaily v případě registrace nového člena a resetu hesla.

Zjištěné nedostatky:

- Zasílané emaily nepodporují diakritiku.
- Email zasílaný při registraci je až příliš stručný (obsahuje pouze *„Vítejte v systému RUPS, vaše přihlašovací údaje jsou: …“*), v důsledku čehož jej většina uživatelů ignoruje.
- Jelikož emaily nejsou zasílány přes řádně nakonfigurovaný SMTP server, končí většina zasílaných emailů ve SPAMu.
- Systém nezasílá další důležité emaily s informacemi, např. při přijetí platby členského příspěvku, varování o neuhrazeném členském příspěvku apod.

[12] CRON – softwarový daemon sloužící pro automatizované spouštění procesů ve stanovených intervalech.

2.2.3 Implementace

Tato kapitola bude pro lepší přehlednost rozdělena na *frontend* a *backend* část. *Frontend* je napsán za použití technologií HTML[13], CSS[14] a JS[15] a tvoří tenkého klienta, který běží v prohlížeči klienta. *Backend* je napsán v jazyce PHP 5.2 a tvoří serverovou část aplikace s MySQL databází.

Komunikace mezi *frontendem* a *backendem* je realizována v podobě AJAX[16] požadavků, kdy *frontend* volá příslušný PHP skript s POST[17] parametry a *backend* pak v případě úspěchu vrací zpět odpověď v JSON[18] formátu.

2.2.3.1 Frontend

Grafické rozhraní je napsáno za pomocí ExtJS frameworku [3] a svým vzhledem připomíná plochu operačního systému Windows. K dispozici je rovněž panel aplikací s nabídkou start. Jednotlivé funkce systému, jako je např. vyhledávání uživatelů, odesílání emailů apod., se pak chovají jako aplikace a při jejich spuštění se vykreslí okno s obsahem a dostupnou funkcionalitou. S okny pak lze pracovat stejně jako v běžném operačním systému, tj. lze je přesouvat, zavřít, minimalizovat na panel aplikací nebo maximalizovat přes celou obrazovku.

Výhodou tohoto řešení je, že si registrátor může souběžně vedle sebe otevřít několik aplikací nebo karet s informacemi o členech a nemusí tak proklikávat mezi jednotlivými stránkami, jak je běžně u webových aplikací zvykem.

Nevýhodou tohoto řešení pak je jeho nepoužitelnost pro mobilní zařízení, což se postupem času ukazuje jako stále větší nedostatek, jelikož registrátor a administrátor nemohou spravovat systém v terénu přes mobilní telefon a pohotově tak reagovat na případné problémy. Nový systém by proto měl tento problém eliminovat.

Jelikož jsou některé pokročilejší funkce schovány přímo v nabídce start (např. funkce pro manuální přiřazení příspěvku stejně jako funkce pro odhlášení), jsou většinou uživatelů tyto funkce přehlíženy, což vede také k tomu, že se uživatelé ze systému odhlašují pouze výjimečně.

Problémem je také dnes již zastaralá knihovna ExtJS, v důsledku čehož nejsou některé komponenty vykreslovány správně. Typickým příkladem je pak chybné vykreslování dialogových oken, které se vykreslují pod otevřenými okny. Uživatel tak musí nejprve všechna okna přesunout na stranu nebo minimalizovat a až následně může zavřít dialogové okno. Vzhledem k tomu, že některá dialogová okna je potřeba před další činností nejprve odsouhlasit a uzavřít, je tato drobná chyba značně nepříjemná.

Posledním zjištěným nedostatkem je pak již dříve zmíněná chybná validace některých formulářových prvků.

2.2.3.2 Backend

Backend aplikace je napsán pro PHP 5.2. V době psaní této práce je již běžně používáno PHP 7.1, které v sobě obsahuje záplaty na řadu bezpečnostních chyb předešlých verzí. Z bezpečnostních důvodů by tak nová verze systému měla být kompatibilní minimálně s PHP 5.6, ideálně pak s PHP 7.1, nebo napsána v jiném programovacím jazyce.

Dalším bezpečnostním problémem je pak to, že je systém napsán v čistém PHP bez použití některého z dostupných PHP frameworků, což v praxi většinou vede k nedostatečné nebo opomenuté sanaci vstupních dat a dalším, převážně bezpečnostním, chybám. Výjimkou z tohoto pravidla bohužel není ani informační systém RUPS. Při analýze a testování informačního systému bylo zjištěno, že je

---

[13] HTML – HyperText Markup Language, jazyk pro tvorbu webových stránek.
[14] CSS – Cascading Style Sheet, jazyk pro popis způsobu zobrazení HTML elementů.
[15] JS – JavaScript, interpretovaný, objektově orientovaný skriptovací jazyk.
[16] AJAX - Asynchronous JavaScript and XML, používá se pro asynchronní komunikaci klienta s webovým serverem.
[17] POST – jedna z dotazovacích metod, pomocí které klient komunikuje s webovým serverem.
[18] JSON – JavaScript Object Notation, datový formát nezávislý na použité platformě.

systém otevřený k nejrůznějším druhům útoků, jmenovitě pak z těch nejběžnějších jde o útoky SQL Injection, XSS a CSRF [4].

Problémová je rovněž validace formulářových prvků, která je z většiny implementována pouze na straně klienta a při znalosti struktury dotazu tak lze serveru podstrčit chybná nebo závadná data.

Vzhledem k uvedeným bezpečnostním nedostatkům byl přístup do informačního systému omezen pouze na veřejné IP adresy klubu HK, z čehož tak plynou další omezení z hlediska vzdálené správy.

2.2.4 Další zjištěné nedostatky

2.2.4.1 Správa fitcentra

Klub HK nabízí svým členům i možnost bezplatného využívání fitcentra HK. Evidence členů fitcentra je v současné době vedena papírově, což je nepraktické a na přelomu zúčtovacího období tak musí správce fitcentra ručně odfiltrovat již neaktivní členy.

Nový systém by proto měl umožňovat vést evidenci i tohoto typu členství a zároveň obsahovat novou roli *Správce fitcentra*, která správci umožní vidět potřebné informace. Zároveň by měl systém registrovat pro správce fitcentra email alias *fitcentrum@hk.cvut.cz*, pro snazší komunikaci se členy.

2.2.4.2 Představenstvo

Členové představenstva klubu HK by měli mít přístup k informacím o stavu členství členů a k výpisu z banky. Ve stávajícím systému ale nelze tyto členy nijak rozlišit, a proto by měl nový systém zahrnovat i novou roli *Člen představenstva* a *Předseda*.

Pro tyto role pak bude informační systém registrovat email aliasy *predstavenstvo@hk.cvut.cz* a *predseda@hk.cvut.cz*.

2.3 Zhodnocení

Analýzou stávajícího řešení bylo zjištěno, že počítačová síť i informační systém obsahují značné funkční a bezpečnostní nedostatky. Jako možné řešení se tak nabízí aktualizace a oprava systému, nicméně vzhledem k počtu a závažnosti zjištěných nedostatků a k zastaralosti celého sytému a použitých technologií se toto řešení nejeví jako ekonomické.

Z tohoto důvodu bude nad rámec této práce navrhnuto nové řešení počítačové sítě, jež bude v krátkosti představeno v následující kapitole, a dále bude navrhnut nový informační systém, který odstraní všechny zjištěné nedostatky.

V závěru této kapitoly je potřeba poznamenat, že i přes velmi ostrou kritiku stávajícího řešení značná část z uvedených problémů v době vývoje a nasazení systému neexistovala. Tyto problémy vznikly postupem času v důsledku příchodu nových technologií, změny politiky klubu HK a také kvůli neodborným zásahům do kódu informačního systému.

Informační systém RUPS i přes všechny jeho nedostatky prokazatelně zjednodušil a zpřehlednil správu členů klubu HK a snížil časové nároky kladené na registrátory a administrátory systému.

3. Návrh nové počítačové sítě

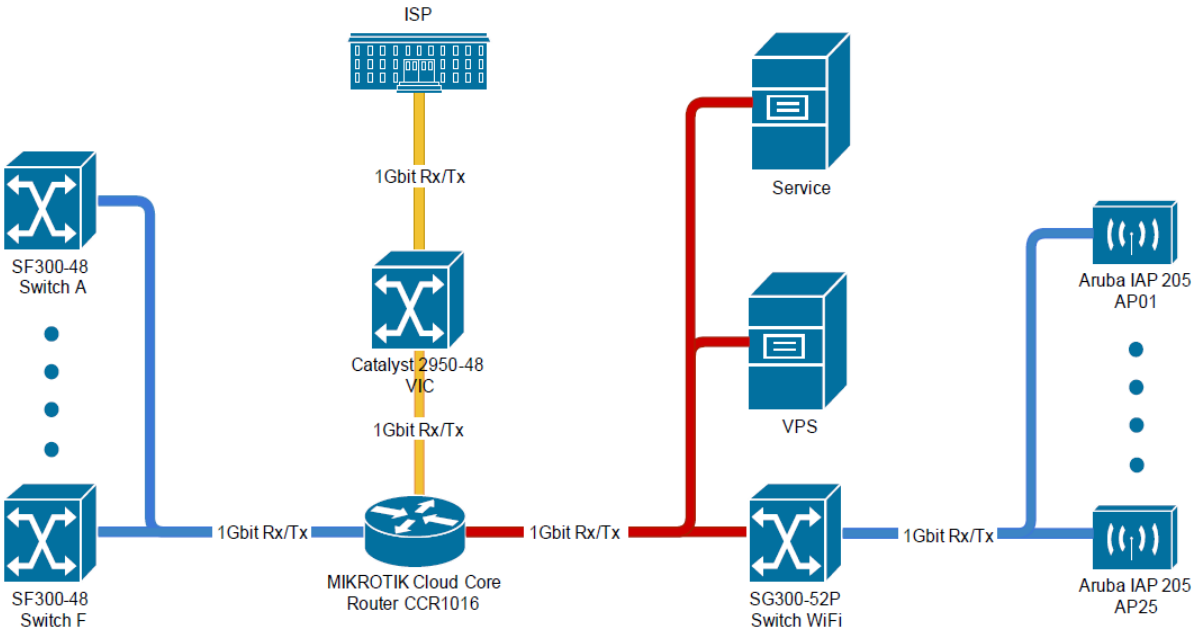
Pro přístup do globální sítě Internet využívá klub akademickou síť ČVUT a CESNET2 a musí se proto řídit pravidly a předpisy o užívání počítačových sítí těchto organizací, ze kterých (mimo jiné) vyplývá, že přístup do počítačových sítí uvedených organizací může být udělen pouze studentům, vědeckým pracovníkům a zaměstnancům ČVUT pro účely studia a vědeckého výzkumu.

Z těchto důvodů je potřeba zamezit přístupu do počítačové sítě klubu HK všem ubytovaným a dalším osobám, které dle uvedených předpisů nemohou tyto počítačové sítě využívat, a dále být schopen v případě porušení nařízení zmíněných organizací (nejčastěji z důvodu sdílení autorsky chráněných děl pomocí peer-to-peer sítí) dohledat viníka a zamezit dalšímu porušování těchto pravidel a nařízení.

Aby bylo možné těmto požadavkům dostát, bylo potřeba navrhnout novou infrastrukturu kolejní sítě a nový informační systém s ohledem na tyto požadavky. Proto bude v této kapitole nejprve v krátkosti představena infrastruktura počítačové sítě, ze které vyplynou další funkční a nefunkční požadavky na funkcionalitu nového systému.

3.1 Infrastruktura počítačové sítě

Klub HK disponuje jedním veřejným blokem IP adres 147.32.98.0/24, který klubu, společně s internetovým připojením, poskytuje VIC ČVUT.



Obrázek 2 – Zjednodušená topologie nové infrastruktury počítačové sítě

Nové schéma zapojení síťových prvků demonstruje Obrázek 2. Ze schématu lze vyčíst, že na kolej je přivedena 1Gbps konektivita, která ústí do switche od VIC ČVUT, na který bude dále napojen router, který bude sloužit jako firewall a dělat NAT překlady IP adres.

Z routeru bude rozvedena 1Gbps páteřní síť mezi jednotlivé patrové *Switche A – F*, které poslouží k metalickému připojení zařízení členů, POE[19] *Wi-Fi Switch*, na který budou napojeny Wi-Fi AP a *Servisní* a *virtualizační* (VPS) servery.

Na serverech *Service* a *VPS* bude nainstalován virtualizační systém Proxmox [5], na kterém pak běží v kontejnerech různé síťové služby, jako například *DHCP, DNS, MX, RADIUS*[20] a *VPN*[21],

[19] POE – Power Over Ethernet, technologie napájení síťových zařízení přes síťový kabel.

přičemž každá důležitá služba bude plně redundantní. *Master* instance služby pak poběží na *Servisním* serveru a *slave* instance na *VPS*.

Díky tomuto modelu bude síť plně redundantní a neomezí tak uživatele v případě havárie nebo odstávky jednoho ze serverů. Do budoucna je pak v plánu pořídit i záložní router, který v případě výpadku hlavního převezme veškerý síťový provoz a zajistí tak nepřetržité připojení k síti Internet.

3.2 Zabezpečení počítačové sítě

Aby bylo možné umožnit každému členovi připojit více zařízení do sítě a zároveň přitom dodržet předpisy všech organizací, bylo potřeba navrhnout nový systém zabezpečení počítačové sítě.

Nově bude mít každý uživatel (člen) přiřazenou vlastní privátní VLAN[22] síť, do které si tak bude moci teoreticky připojit až 253 koncových zařízení. Standardně se přitom počítá zhruba se třemi zařízeními na uživatele (počítač, chytrý telefon a tablet).

Komunikace směrem do internetu pak bude probíhat z této sítě přes veřejnou IP adresu, která je přiřazená k této privátní VLAN síti. Díky tomuto modelu pak lze jednoznačně dohledat a identifikovat člena, který se dopustil porušení předpisů na základě jeho veřejné IP adresy, a to jak na metalickém, tak na bezdrátovém připojení.

Pro účely autentizace přístupu do sítě bude využit RADIUS server, který pomocí protokolu IEEE 802.1X[23] vyzve uživatele, aby poskytl validní přihlašovací údaje od svého klubového účtu, a zároveň se provede kontrola MAC adresy připojovaného zařízení.

Aby se tedy mohl uživatel úspěšně připojit do počítačové sítě, musí být registrovaným členem klubu, mít aktivní členství, aktivní službu síť a musí mít v systému registrovanou MAC adresu zařízení, přes které se chce do sítě připojovat.

Switch nebo Wi-Fi AP pak na základě odpovědi z RADIUS serveru připojí uživatele do VLAN sítě na základě těchto pravidel:

1) V případě, kdy uživatel splnil všechny podmínky na připojení (tj. poskytl správné autentifikační údaje, má aktivní členství a službu síť a připojuje se ze zařízení, které je evidované u jeho účtu), je připojen do jeho registrované VLAN sítě.
2) V případě, kdy uživatel splnil všechny podmínky z bodu 1), ale snaží se připojit ze zařízení, které není evidováno u jeho účtu, je připojen do sítě *vlan-unknown-device*.
3) V případě, kdy uživatel poskytl správné autentifikační údaje, má aktivní členství, ale nemá aktivní službu síť z důvodu trestu za porušení podmínek této služby, je připojen do sítě *vlan-ban*.
4) V případě, kdy uživatel poskytl správné autentifikační údaje, ale nemá aktivní členství nebo službu síť, je připojen do sítě *vlan-inactive*.
5) V ostatních případech je pak uživatel připojen do sítě *vlan-guest*.

V bodech 2) až 5), kdy uživatel nesplní některou z podmínek připojení, bude připojen do počítačové sítě a může přistupovat na služby a aplikace na doméně *hk.cvut.cz*. Zároveň mu ale bude odepřen přístup do sítě Internet. V bodech 2) až 4) bude navíc uživatel po připojení přesměrován na stránky informačního systému, kde systém (na základě IP adresy VLAN sítě) zobrazí informace s odůvodněním, proč byl uživateli zamezen přístup do sítě Internet.

Pokud bude uživatel připojen do *guest* sítě, bude přesměrován na stránky *hk.cvut.cz*, na kterých se může dočíst, jak má dále postupovat, a nastavit svá zařízení v případě, že chce získat plnohodnotný přístup k internetu (což bude užitečné hlavně pro nově ubytované studenty na koleji).

---

[20] RADIUS – Služba pro autentifikaci a autorizaci uživatelů.
[21] VPN – Virtual Private Network, prostředek pro propojení počítačů prostřednictvím veřejné (nedůvěryhodné) počítačové sítě.
[22] VLAN – Virtuální LAN síť, používá se k segmentaci LAN sítě bez potřeby dodatečného HW.
[23] IEEE 802.1X – protokol umožňující vyšší zabezpečení počítačové sítě.

Tento model tedy bude umožňovat několika-fázovou autentifikaci uživatele. Navíc budou porty switchů konfigurovány dynamicky na základě odpovědi z RADIUS serveru, díky čemuž se uživatel může připojit odkudkoliv na koleji a nebude tak vázán na konkrétní zásuvku, jako je tomu u stávajícího řešení (viz kapitola 2.1).

Pro síťové prvky a služby bude zároveň vytvořena VLANa *vlan-service*, do které budou tato zařízení posazena, a do které bude mít přístup pouze administrátor systému. Na routeru pak budou podle potřeby nastaveny Dst-NAT pravidla pro porty jednotlivých služeb tak, aby byly dosažitelné.

3.3 Požadavky na systém

Díky tomuto rozvržení bude potřeba jednorázově nakonfigurovat router, patrové switche, Wi-Fi AP a DHCP server a nebude tak potřeba pro tyto prvky sítě generovat aktualizovanou konfiguraci.

Služby MX, RADIUS, VPN, PAM a další pak budou napojeny přímo na databázi nového systému, kterou tak bude potřeba k těmto účelům navrhnout.

Jedinou výjimkou bude generovaný seznam MAC adres zařízení, které budou mít v RADIUSu povolenou funkci MAC-bypass (viz kapitola 4.4).

3.4 Zhodnocení

Nová počítačová síť byla navrhnuta s ohledem na aktuální potřeby klubu HK, jeho finanční prostředky a s ohledem na vlastní dostatečnou technologickou znalost problematiky počítačových sítí.

Navrhované řešení sice nemusí být optimální, a s největší pravděpodobností nepůjde aplikovat na větší koleje (především z důvodu potřeby velkého množství VLAN a veřejných IP adres), nicméně pro stávající potřeby a možnosti klubu HK je více než dostačující a zároveň odstraní všechny zjištěné nedostatky stávajícího řešení.

Celý koncept nové sítě byl mnohokrát diskutován se správci počítačové sítě klubu Silicon Hill a dalšími osobami, které se aktivně zabývají touto problematikou.

Systém byl zároveň navrhnut tak, aby veškerá konfigurace síťového HW a drtivé většiny služeb proběhla pouze jednorázově a dodatečné informace o přístupu ke službám byly získávány primárně z databáze nového systému, tj. bez potřeby upravovat konfiguraci HW a služeb ať už manuálně, či prostřednictvím informačního systému. V důsledku toho bude celá síť méně náchylná k potenciálním chybám a bude vyžadovat minimální, ideálně pak žádnou, správu.

4. Funkční požadavky

V této kapitole budou nejprve podrobně představeny uživatelské role systému spolu s výčtem jejich pravomocí a úloh, které budou vykonávat.

Následně budou představeny ucelené požadavky na evidenci členů klubu, evidenci přijatých členských příspěvků, požadavky pro služby *Počítačová síť* a *Fitcentrum HK* a další funkční požadavky celého systému.

4.1 Uživatelské role

Chod klubu a provoz jeho služeb obstarává několik aktivních jedinců z řad jeho členů. Konkrétně se pak jedná o *Předsedu klubu*, *kolejní radu* (nebo též *představenstvo*), *registrátora*, *správce počítačové sítě* a *správce fitcentra HK*, přičemž každý z těchto subjektů má pevně definované pravomoci a povinnosti. Člen klubu pak může být pověřen výkonem jedné či více z uvedených funkcí.

Toto rozložení funkcí a jejich pravomoci by měl částečně reflektovat i nový systém na základě výčtu dostupných uživatelských rolí. Konkrétně se pak bude jednat o role *administrátor, předseda, člen rady*, *registrátor*, *správce sítě*, *správce fitcentra* a *člen*, kde role *člen* je implicitně přiřazená každému členovi klubu. Systém bude rovněž umožňovat přiřazení více rolí (a tedy i funkcí) jednomu uživateli.

4.1.1 Administrátor

- Má pravomoci k veškeré dostupné funkcionalitě systému.
- Může přiřazovat dalším uživatelům systémové role.
- Může pozastavit nebo zrušit členství, k čemuž může dojít z důvodu opakovaného porušování stanov a interních předpisů klubu nebo předpisů nadřazených.
- Může individuálně upravit datum splatnosti členského příspěvku nebo členovi udělit za významnou činnost pro klub *čestné členství*. V takovém případě je pak dotyčnému členovi zcela prominuta povinnost hradit členské příspěvky.
- Může členovi udělit odpustek platby členského příspěvku za vykonanou práci pro klub. V takovém případě je pak dotyčnému členovi jednorázově prominuta povinnost uhradit členský příspěvek. Administrátor by měl (nepovinně) rovněž u odpustku poznamenat důvody, které vedly k jeho udělení.
- V případě, že systém nebyl schopen automaticky přiřadit došlou úhradu členského příspěvku (např. na základě špatného variabilního symbolu), musí mít administrátor možnost manuálně přiřadit platbu ke členovi na základě dostupných informací o platbě.
- Může upravovat parametry nastavení systému (nastavovat zúčtovací období apod.).

4.1.2 Předseda

Má stejné pravomoci jako administrátor systému. Nemá však k dispozici některé pokročilejší funkce systému nebo všechny možnosti nastavení služeb. Konkrétně pak nemůže:

- Měnit uživatelské jména.
- Spravovat pokročilé nastavení služeb (např. přiřadit konkrétní VLAN síť atd.).
- Nemá přístup k nastavení informačního systému.

4.1.3 Člen rady

- Má přístup ke všem informacím členů klubu v systému, nemá ale pravomoci tyto informace jakkoliv měnit a upravovat.
- Má přístup k výpisu z bankovního účtu, nemá ale pravomoci tyto údaje jakkoliv měnit, nebo proběhlé transakce párovat s účty členů klubu.

4.1.4 Registrátor

- Registruje nové členy do klubu, je pověřený správou evidence členů klubu a může editovat jejich veškeré osobní informace.
- Na žádost člena je oprávněn změnit jeho přístupové heslo do systému.
- Může členovi povolovat a zakazovat veškeré služby a je oprávněn zobrazit si jejich základní nastavení a historii trestů (za porušení pravidel poskytování služby), není oprávněn ale tyto informace upravovat. Toto oprávnění má pouze administrátor, předseda nebo správce příslušné služby. Výjimkou je správa registrovaných zařízení u služby počítačová síť, kde je registrátor oprávněn upravovat evidenci registrovaných zařízení člena.
- Má přístup k seznamu evidovaných plateb a odpustků člena, nemůže ale tyto informace jakkoliv měnit.
- V případě potřeby může vytisknout registrační formulář nebo složenku pro úhradu členského příspěvku.

4.1.5 Správce sítě

- Má přístup k osobním informacím členů, není ale oprávněn tyto informace měnit.
- Na žádost člena je oprávněn změnit jeho přístupové heslo nebo uživatelské jméno v systému.
- Má veškeré pravomoci týkající se nastavení služby *počítačová síť*, tj. individuální povolení nebo zakázání služby členovi, správa evidence registrovaných zařízení člena a historie jeho trestů za porušení pravidel a nařízení týkajících se této služby.

4.1.6 Správce fitcentra

- Má přístup k osobním informacím členů, není ale oprávněn tyto informace měnit.
- Má veškeré pravomoci týkající se nastavení služby *fitcentrum HK*, tj. individuální povolení nebo zakázání služby členovi a historie jeho trestů za porušení pravidel a nařízení týkajících se této služby.

4.1.7 Člen

- Je oprávněn zobrazit si veškeré informace o něm vedené v systému, mimo jiné také informace o nastavení jeho služeb a informace o jeho evidovaných úhradách členských příspěvků.
- Uživatel je oprávněn změnit si kontaktní email, telefonní číslo a heslo používané k přihlášení do systému a ostatním službám.

4.2 Evidence členů

Aplikace bude umožňovat vést a spravovat evidenci členů klubu. U každého člena je potřeba povinně evidovat *jméno, příjmení, datum* a *místo narození, adresu trvalého bydliště* a *email,* nepovinně pak *telefonní číslo* a případně další informace, např. v podobě poznámky.

Dále bude systém evidovat *datum registrace člena, datum následující platby* členského příspěvku, informaci, zda bylo členovi uděleno *čestné členství* a jeho *přidělené role*.

Systém by rovněž měl umožnit zobrazit rychlý přehled aktivovaných služeb člena a jejich nastavení.

4.2.1 Uživatelské jméno

Každý člen má navíc v systému unikátní uživatelské číslo tzv. *UID*. UID je čtyřmístné celé číslo, automaticky přidělené systémem při registraci, které slouží pro potřeby identifikace člena a jeho příchozích plateb (viz kapitola 4.3).

Při registraci systém vygeneruje členovi uživatelské jméno na základě těchto pravidel:

1) Uživatelské jméno bude složeno ze jména a příjmení člena oddělených tečkou.
2) Pokud má uživatel více jmen nebo příjmení (většinou zahraniční studenti), budou tato jména oddělena podtržítkem.

3) V uživatelském jménu budou nahrazena písmena obsahující diakritiku za jejich ASCII alternativu, vynechány další interpunkční symboly a velké znaky budou převedené na malé.
4) V případě shody takto vygenerovaného uživatelského jména s již existujícím uživatelem v systému bude na konec přidáno číslo, které zajistí jeho unikátnost, přičemž se čísluje od čísla 2.

Příkladem správně vygenerovaného jména je například *hai_yen.luong2* z původního jména Luöng, Hai Yen. Uživatelské jméno obsahuje na svém konci číslo 2, jelikož v systému již existuje uživatel s tímto jménem.

Uživatelské jméno je oprávněn editovat pouze *administrátor* nebo *správce sítě*.

4.2.2 Registrace nového člena

Nové členy je oprávněn registrovat pouze *registrátor*, *předseda* nebo *administrátor* systému, který při registraci vyplní všechny povinné osobní informace a nastaví uživateli jazyk aplikace (čeština nebo angličtina).

Před registrací nového člena by měl systém automaticky prohledat evidenci stávajících členů a v případě zjištění možné duplicity upozornit na tuto duplicitu před samotným založením nového účtu člena. Kontrola existence člena v systému bude probíhat na základě fulltextového vyhledávání nad jeho jménem a příjmením a shodě data narození.

V případě, že systém nedetekuje možnou duplicitu, nebo registrátor ověří, že se jedná o jinou osobu, bude členovi založen nový účet v informačním systému.

4.2.3 Vyhledávání členů

Systém bude umožňovat vyhledávat nad členy klubu (stávajícími i bývalými), a to minimálně podle *UID, jména, příjmení, data narození* a *IP adresy*. Vyhledávání nesmí být citlivé na velikost znaků.

Ve výsledcích bude uveden stav členství a služeb (aktivní, neaktivní, ban).

4.3 Evidence příchozích a odchozích plateb

Jako hlavní zdroj příjmů klubu slouží dary a členské příspěvky, které jsou povinni hradit všichni členové klubu každý kalendářní půlrok, tj. od 1.1 a 1.7., přičemž doba splatnosti (a také doba, do kdy zůstávají členovi aktivní služby) je podle aktuální vyhlášky stanovena na 28. 2. a 30. 9. V praxi to tedy znamená, že člen může uhradit členský příspěvek za první půlku roku kdykoliv od 1. 1. do 28. 2. a za druhou půlku pak kdykoliv od 1. 7. do 30. 9.

Dále má dle aktuální vyhlášky člen povinnost uhradit první členský příspěvek nejpozději do 7 kalendářních dnů po registraci v systému. Po uplynutí těchto lhůt je členovi pozastaveno členství v klubu (a tedy i provedena deaktivace služeb), a to až do doby úhrady členského příspěvku za aktuální kalendářní půlrok.

Z této platební politiky tedy plynou další požadavky na informační systém.

4.3.1 Nastavení systému plateb

Administrátor bude mít možnost upravovat parametry nastavení systému plateb, konkrétně pak půjde o tyto parametry:

- Nastavení bezplatného období, tj. kolik dní má člen na uhrazení prvního členského příspěvku po registraci do klubu.
- Výši a měnu členského příspěvku společně s jeho maximální možnou odchylkou (v některých případech dochází k drobným odchylkám z důvodu změny kurzu při úhradě členského příspěvku v jiné měně) a číslo bankovního spojení.
- Začátek prvního a druhého zúčtovacího období společně s dobou splatnosti.
- Případné další parametry.

4.3.2 Stahování příchozích a odchozích plateb

Systém by měl být schopen periodicky stahovat nové bankovní transakce z bankovního účtu, a to ideálně několikrát v průběhu dne.

V případě chyby, např. z důvodu ztráty spojení nebo změny vystavovaného API banky, aplikace nesmí havarovat a musí o této události informovat správce systému (*administrátora*), např. prostřednictvím emailu, který následně situaci zhodnotí a v případě potřeby provede nezbytné kroky k nápravě (např. požádá vývojáře systému o úpravu rozhraní API).

4.3.3 Automatické párování plateb

Nově příchozí platby budou na základě shody variabilního symbolu uvedeného u platby s UID člena automaticky párovány s jeho účtem. Automaticky párované platby musí mít částku v měně CZK, a dále musí být částka v rozsahu, který je nastaven v systému *administrátorem*.

Pokud nově příchozí platba nebude odpovídat parametrům nastaveným v systému, bude o této skutečnosti informován *administrátor* systému (např. prostřednictvím emailu) a platba nebude automaticky spárována.

V případě nové odchozí platby nebude probíhat automatické párování s účtem člena.

4.3.4 Manuální párování plateb

Pro případy, kdy systém není schopen automaticky přiřadit příchozí platbu k účtu člena, nebo platba neodpovídá nastaveným parametrům v systému, musí mít možnost toto propojení manuálně vytvořit *administrátor*. Pro tyto účely by měl mít *administrátor* k dispozici veškeré dostupné informace o platbě, které lze z API bankovního účtu získat.

Pro případ chybného automatického párování (např. v důsledku chybně uvedeného variabilního symbolu) musí mít *administrátor* prostředky pro zrušení této vazby nebo pro přiřazení platby k jinému členovi.

4.3.5 Odpustky

Vykoná-li člen během kalendářního půlroku záslužnou práci pro klub, může mu být na základě rozhodnutí *kolejní rady* udělen odpustek, tj. prominutí povinnosti uhradit členský příspěvek v následujícím kalendářním půlroce.

Systém proto musí umožnovat udělování odpustků členům a rovněž v případě potřeby (například lidské chyby) odpustek odebrat.

Z důvodu lepšího celkového přehledu nad členskými příspěvky budou odpustky zahrnuty do výpisu ostatních plateb, přičemž budou od ostatních transakcí nějak odlišeny (např. podbarvením nebo sloupcem s typem platby).

4.3.6 Určení data další úhrady příspěvku

Na začátku kapitoly 4.3 byla popsána data splatnosti členských příspěvků. Systém ale musí umožňovat manuálně upravit toto datum splatnosti v případě individuální žádosti člena po schválení *kolejní radou*.

Typickým příkladem je například prodlení s platbou nebo předčasný nástup studenta na kolej (myšlen nástup více jak 7 kalendářních dní před začátkem nového kalendářního půlroku), v důsledku čehož by člen musel uhradit členský příspěvek dvakrát během krátké doby.

V ideálním případě by měl systém přepočítávat datum splatnosti individuálně pro každého člena podle následujících pravidel:

1) V případě automaticky spárované příchozí platby nebo udělení odpustku systém upraví datum splatnosti člena na následující zúčtovací období, tj. na dobu splatnosti následujícího kalendářního půlroku, je-li tato hodnota větší než již nastavené datum další splatnosti členského příspěvku člena.

2) V případě manuálního odebrání přiřazené platby nebo zrušení odpustku bude zjištěn poslední kalendářní půlrok, za který měl člen uhrazený příspěvek nebo udělený odpustek, a datum další splatnosti se nastaví na konec splatnosti následujícího kalendářního půlroku.

4.3.7 Aktivace a deaktivace členství

Pokud člen nestihne uhradit členský příspěvek do data splatnosti a nepožádá si o jeho prodloužení, nebo nebude této žádosti vyhověno, provede systém pozastavení členova účtu a deaktivaci všech jeho služeb, tj. zamezení jejich dalšího využívání. Uživateli však stále zůstává plný přístup do systému na základě jemu přidělených rolí.

Opětovná aktivace uživatelova účtu je pak možná pouze v případě:

1) uhrazení členského příspěvku;
2) udělení odpustku;
3) prodloužení data splatnosti členského příspěvku.

Pokud dojde k splnění některé z výše uvedených podmínek, měl by systém automaticky provést reaktivaci členova účtu a všech jeho služeb automaticky. V opačném případě nesmí systém reaktivaci členova účtu povolit.

4.4 Požadavky počítačové sítě

Z návrhu infrastruktury nové počítačové sítě vyplývá, že pro zaručení správného fungování sítě a splnění všech jejich požadavků musí systém implementovat další dodatečnou funkcionalitu, kterou si níže představíme podrobněji.

4.4.1 Evidence zařízení

Informační systém musí umožňovat vést u každého člena evidenci jeho zařízení, se kterými je oprávněn připojovat se do počítačové sítě klubu. Tato evidence bude spravovat informace o MAC adrese zařízení, názvu zařízení a dále možnost povolit nebo zakázat MAC-bypass.

V případě povolení MAC-bypass bude uživateli umožněn přístup do počítačové sítě pouze na základě MAC adresy zařízení, tedy bez potřeby navíc uvádět přístupové údaje do informačního systému klubu. Důvodem této funkcionality je případ, kdy uživatel potřebuje připojit do počítačové sítě zařízení (např. router), které nepodporuje protokol IEEE 802.1X.

V defaultním nastavení bude MAC-bypass deaktivován a v případě potřeby bude povolován na základě oprávněné žádosti pouze u konkrétních zařízení.

Jelikož se aktivita uživatele bude v případě potřeby dohledávat na základě VLAN sítě přiřazené k uživateli, do které jsou přiřazena veškerá jeho zařízení, není potřeba vést historii evidence registrovaných zařízení.

4.4.2 Historie trestů

U služby počítačová síť bude vedena historie trestů (BANů), které uživatel obdržel za porušení pravidel o využívání počítačové sítě, kde bude moci *správce sítě*, *předseda* nebo *administrátor* přidávat, editovat a rušit tresty.

U trestu bude veden jeho začátek a konec (ve dnech) a dále bude možné k němu přidat textové odůvodnění, proč byl tento trest členovi udělen.

Systém bude rovněž schopen na základě uděleného trestu, automaticky deaktivovat síťové členství a po jeho vypršení opět zpětně aktivovat síťovou službu.

Ve výpisu trestů bude opticky rozlišen aktuálně probíhající trest, například podbarvením řádku v případě tabulkového výpisu.

4.4.3 Nastavení VLAN sítě

Při aktivaci síťového členství, přiřadí členovi systémem automaticky VLAN síť. VLAN síť bude vybrána podle následujících pravidel:

1) Pokud některá z uživatelských VLAN není používána, bude členovi přiřazena ta VLAN síť, která byla po nejdelší dobu nepoužívána.
2) Pokud nebude volná žádná VLAN síť, bude uživatel přiřazen do VLAN sítě, kterou má v současné době přiřazeno nejméně členů.

Díky tomuto nastavení bude sníženo riziko, že na jedné VLANě dojde během krátké doby opakovaně k porušení pravidel týkajících se peer-ro-peer sítí a tedy, že nebude v krátké době CESNETem zaslána opakovaná stížnost na stejnou veřejnou IP adresu.

Dále v případě, kdy dojde k uvolnění některé z VLAN, systém zkontroluje, zda v současné době není více členům přiřazena stejná VLAN síť (viz bod 2), a pokud bude zjištěno že ano, přiřadí jednomu ze členů nově uvolněnou VLAN síť. Za optimálního provozu by se pak nemělo stát, že dva uživatelé budou mít přiřazenou stejnou VLAN síť, a tedy i stejnou veřejnou IP adresu.

Navíc systém v případě potřeby umožní *administrátorovi* nebo *správci sítě* manuálně vynutit nastavení konkrétní VLAN sítě členovi.

4.4.4 Historie přiřazených VLAN sítí

Systém rovněž umožní vypsat kompletní historii přiřazených VLAN sítí společně s časem, od kdy do kdy byly uživatelovi přiřazeny.

Systém by měl zároveň nějak opticky rozlišit členovu aktuálně přiřazenou VLAN síť.

4.4.5 Landing pages

V případě, kdy bude členovi zakázán přístup do sítě internet, tj. bude se nacházet ve VLAN síti *vlan-inactive, vlan-ban, vlan-unknown-device* nebo *vlan-guest,* provede počítačová síť přesměrování všech odchozích HTTP požadavků na informační systém, který pak na základě IP adresy VLAN sítě zobrazí informace o důvodu, proč byl uživateli odepřen přístup do sítě internet.

V případě, kdy se bude uživatel nacházet v síti *vlan-guest*, provede systém přesměrování na internetové stránky klubu HK.

4.5 Požadavky fitcentra HK

4.5.1 Evidence vstupů do fitcentra

Nově se také uvažuje o propojení informačního systému se systémem vrátnice tak, aby vrátní nemuseli vést evidenci vstupů do fitcentra v papírové podobě. Nový systém by tak měl umožnovat vést evidenci vstupů a odchodů z fitcentra HK. Zároveň by mel vystavovat REST API, pro možnost budoucího propojení s aplikací vrátnice.

4.5.2 Historie trestů

U služby fitcentrum HK bude vedena historie trestů, které uživatel obdržel za porušení pravidel o využívání fitcentra HK, kde bude moci *správce fitcentra*, *předseda* a *administrátor* přidávat, editovat a rušit tresty.

U trestu bude veden jeho začátek a konec (ve dnech) a dále bude možné k němu přidat textové odůvodnění, proč byl tento trest členovi udělen.

Systém bude rovněž schopen na základě udělených trestů členovi automaticky deaktivovat a aktivovat členství (po jeho vypršení, smazání apod.).

Ve výpisu trestů bude opticky rozlišen aktuálně probíhající trest, například podbarvením řádku v případě tabulkového výpisu.

4.6 Maily

Jak bylo zjištěno při analýze, zasílané emaily jsou jedním z největších problémů stávajícího řešení. Proto budou nově nakonfigurovány dva MX servery, které budou zároveň podporovat zabezpečený SMTP protokol.

Nový systém zároveň musí podporovat v emailech diakritiku a pro emaily by měla být ideálně vytvořena přehledná HTML šablona, která příjemce dostatečně zaujme natolik, aby si email přečetl.

Zároveň by měl nový systém zasílat minimálně tato oznámení:

- oznámení o registraci společně se zadanými přístupovými údaji a informacemi pro uhrazení prvního členského příspěvku spolu s datem splatnosti;
- oznámení potvrzující změnu přístupových údajů (jméno, heslo);
- oznámení o přijetí úhrady členského příspěvku, nebo udělení odpustku;
- varování o blížícím se datu splatnosti členského příspěvku (při dosud nepřijaté platbě). Varování bude zasíláno 28 a 14 dní před datem splatnosti;
- varování o příchozí platbě, kterou se nepodařilo automaticky propojit s účtem člena, nebo která nesplňuje požadavky definované pro automatické propojení s účtem člena, které bude zasáláno *administrátorovi*.

Systém by měl rovněž v případě jakékoliv chyby zaslat *administrátorovi* email s varováním.

4.7 Reporty

V současné době je potřeba generovat pouze report o stavu členství v klubu. Tento report je generován vždy na konci kalendářního půlroku, přičemž obsahuje seznam členů, kteří za tento kalendářní půlrok zaslali členský příspěvek, nebo jim byl v tomto kalendářním půlroce udělen odpustek.

Report je řádkový a u každého člena obsahuje jeho *UID, jméno, příjmení, datum narození, adresu trvalého bydliště, místo narození* a výši uhrazeného nebo prominutého členského příspěvku.

Report bude generován v CSV, XLS, XLSX nebo obdobném formátu a bude možné tento report generovat i zpětně.

4.8 Tisk dokumentů

Systém bude umět na vyžádání vygenerovat a vytisknout členovu přihlášku do klubu a složenku pro uhrazení členského příspěvku.

4.9 Zhodnocení

Na základě analýzy nedostatků stávajícího řešení, požadavků nového řešení počítačové infrastruktury a námětů kolegů byly představeny ucelené funkční požadavky na nový informační systém. Nové požadavky se rovněž opírají o několikaletou osobní zkušenost práce se stávajícím systémem, a tedy i dokonalou znalost všech procesů v rámci klubu a informačního systému.

Nové požadavky definují lepší členění uživatelských rolí, a tedy i lepší řízení přístupu k osobním informacím členů a jejich službám. Správa evidence členů by měla být jednodušší a přehlednější. Veškeré procesy v systému by měly být maximálně automatizované, nicméně stále by měla zůstat možnost manuální úpravy jinak automaticky konfigurovaného nastavení. Důležité jsou rovněž požadavky na ukládání historie síťového nastavení (přiřazených VLAN sítí) s možností jejich zpětného dohledání.

Požadavky na emailový systém by měly zmenšit procento opožděných plateb díky včasně zasílaným připomínkám o úhradě členského příspěvku a zároveň zvýšit reakce schopnost správce systému v případě výskytu chyby systému.

5. Nefunkční požadavky

5.1 Hardware

K dispozici jsou celkem 2 servery, *Service* a *VPS*. Na obou těchto serverech budou pro informační systém vytvořeny Linux kontejnery s těmito parametry:

- CPU: Intel Xeon 2.0 Ghz
- RAM: 2GB
- 1x LAN
- OS: Debian 8 „jessie“

5.2 Přístup k datům

Databáze pro informační systém by měla být zvolena i v závislosti na možnosti napojení na jednotlivé síťové služby, jako je *MX, RADIUS, PAM, OpenVPN* a další, nebo musí systém pro tyto služby vystavovat API.

5.3 Výkon

Jelikož má klub v průměru okolo 200 členů a vzhledem ke kapacitě koleje se neočekává, že by toto číslo do budoucna o moc narostlo, nejsou na výkon systému kladeny moc velké nároky.

V praxi využívá v průměru grafické rozhraní vždy jen jeden uživatel (správce) z důvodu potřeby úpravy nastavení člena nebo kontroly jeho členských příspěvků. Systém by tak měl být schopen obsloužit alespoň 10 požadavků za sekundu. Zároveň bude-li napojení na síťové služby probíhat přes vystavované API, měl by systém dostatečně rychle odpovídat na tyto zasílané požadavky, především pak u RADIUS serveru, který slouží pro autentifikaci uživatelů na počítačové síti, kde by delší prodlevy mezi autentifikačním dotazem a odpovědí mohly znamenat krátké odpojení člena od počítačové sítě.

Vzhledem k počtu uživatelů a počtu jejich zařízení by měl být systém schopen odpovědět alespoň na 20, ideálně pak na 40, autentifikačních požadavků za 1 sekundu.

5.4 Škálovatelnost

Pokud bude výsledný systém dostatečně výkonný, pak vzhledem k relativně malému počtu uživatelů nemusí být horizontálně škálovatelný ve smyslu, že bude zátěž symetricky rozdělena na obě instance Linux kontejnerů. Navržený systém by ale měl být redundantní, tj. v případě výpadku jednoho ze serverů bude funkcionalita automaticky převedena na server záložní.

5.5 Rozšiřitelnost a modifikovatelnost

Systém by měl být napsán v některém z běžných jazyků (Java, PHP, Python) a používat známé a dobře dokumentované frameworky a knihovny tak, aby mohl být potenciálně do budoucna dále vyvíjen i jinými programátory.

Zároveň by měl být systém (v rámci možných mezí) snadno rozšiřitelný o další funkcionalitu.

5.6 Použitelnost

Systém musí být (v rámci přijatelných mezí) jednoduchý na nasazení a údržbu. Bude-li systém ke své funkci vyžadovat instalaci nebo konfiguraci dalších služeb a aplikací, musí být zdokumentována jejich instalace a obsluha v interní Wiki klubu, která slouží především k těmto účelům.

5.7 Bezpečnost

Systém musí být bezpečný proti nejběžnějším druhům útoků, jako je SQL Injection, XSS, CSRF, Session Hijacking a měl by v odpovědi zasílat správně nakonfigurované HTTP hlavičky (Allow-Access-Origin, X-Frame-OPTIONS apod.).

Systém nesmí umožnit neoprávněné osobě přístup k funkcionalitě a datům, ke kterým nemá nastavené právo (rozlišováno podle uživatelovy role).

Samozřejmostí je pak solení a hashovaní hesel minimálně přes algoritmus SHA-2 nebo lepší.

5.8 Zhodnocení

Vzhledem k malému počtu aktivních uživatelů nejsou na systém kladeny vysoké výkonnostní a HW nároky. Nejkritičtějším parametrem je tak doba odezvy na autentifikační požadavek.

Výsledný systém by měl být zároveň dobře rozšiřitelný, jednoduchý na správu a bezpečný.

V následující kapitole se podíváme na již existující informační systémy jiných kolejí ČVUT a zhodnotíme možnost jejich případného využití pro potřeby klubu Hlávkova kolej.

6. Analýza existujících řešení

Požadavky na informační systém klubu HK jsou dosti specifické a z existujících řešení se jim tak nejvíce blíží informační systémy klubů na jiných kolejích ČVUT. Z tohoto důvodu byla vypracována analýza informačních systémů klubů Silicon Hill (kolej Strahov), Pod-o-lee (kolej Podolí), Masařka (Masarykova kolej), Sincoolka (Sinkuleho a Dejvická kolej), Buben (Bubenečská kolej) a Orlík (kolej Orlík).

Vzhledem k rozsahu a obsáhlosti informačních systémů byla provedena pouze okrajová analýza těchto informačních systémů s ohledem na dostupnou funkcionalitu, bezpečnost řešení, náročnost potřebných úprav pro potřeby klubu HK a použitelnost z hlediska nároků na údržbu a správu těchto systémů.

V závěru analýzy každého informačního systému bude provedeno jeho zhodnocení s ohledem na možnost využití systému pro potřeby klubu HK.

6.1 Informační systém klubu Silicon Hill

Informační systém klubu Silicon Hill (Strahovské koleje) IS.sh vytvořili Bronislav Robenek a Dominik Mališ během léta 2012 a do ostrého provozu byl nasazen v 2. polovině roku 2012 [6].

6.1.1 Implementace

Jedná se o plně horizontálně škálovatelný systém, jehož HTML frontend je vytvořen za pomocí frameworku Twitter Bootstrap [7]. Backend je pak napsán v jazyce Ruby on Rails a skládá se z web role, která se stará o generování HTML frontendu a obsluhu požadavků, a worker role, která běží na pozadí a obsluhuje déle trvající operace, typicky pak konfiguraci síťových prvků, stahování bankovních transakcí apod.

K dispozici jsou 2 servery, na kterých běží několik instancí web rolí a worker rolí, přičemž veškerá zátěž je rovnoměrně rozložena na všechny instance pomocí load-balancingu.

Pro potřeby databáze je použita PostgreSQL a OpenDJ (LDAP), na kterých je nastavena master-master replikace. Přímo na tyto databáze je pak jednak napojen informační systém, ale také veškeré síťové služby. Konkrétně se pak jedná např. o služby PowerDNS (DNS), ISC DHCP, dhcpy6d (DHCP pro IPv6), FreeRADIUS, Postfix a další.

6.1.2 Evidence členů a členství

Systém umožňuje online před-registraci členů. Člen se tak může na stránkách informačního systému před-registrovat a následně se pak už jen dostaví za registrátorem, který zkontroluje správnost zanesených informací a jedním klikem vytvoří účet člena v informačním systému. Stále ale zůstává potřeba vyplnit i papírovou verzi registračního formuláře. Kromě povinných osobních informací, které jsou definovány ve Stanovách Studentské unie ČVUT, lze v systému u člena vést i jeho rodné číslo, název školy, kterou člen studuje, a profilovou fotku.

Zajímavostí je, že systém umožňuje vést evidenci i druhů členství členů (*základní členství*, *síťové členství*, *členství ve fitcentru*, *členství v bastlírně* atd.), u kterých lze zároveň definovat, zda je členství zpoplatněno (a případně jeho částku a kód variabilního symbolu), zda je závislé na jiném druhu členství, a délku jeho platnosti.

Samozřejmostí je pak automatická aktivace nebo prodloužení členství při přijetí příspěvku a jeho deaktivace po vypršení jeho doby splatnosti.

6.1.3 Správa počítačové sítě

Informační systém byl (stejně jako ostatní) primárně navržen pro správu členů a řízení jejich přístupu do počítačové sítě a k tomuto účelu má bohatou paletu funkcionality.

Pro lepší správu je v systému vedena evidence tzv. zón. Zóny reprezentují jednotlivé bloky koleje a každému správci a administrátorovi lze omezit přístup pouze na konkrétní zóny.

Informační systém rovněž vede evidenci síťových prvků, společně s výčtem jejich portů, umístěním v zóně a dalšími dodatečnými informacemi (např. kde se prvek fyzicky nachází, do kterého pokoje je vyveden přístupový port, číslo portu atd.).

Dále systém umožňuje evidovat VLAN sítě a určovat, na kterých síťových prvcích budou tyto VLAN sítě dostupné. Společně s VLAN sítí pak lze definovat konfiguraci DHCP serveru a IP rozsahu, který je na VLAN síti dostupný.

U uživatelů počítačové sítě se pak vede seznam jejich zařízení společně s jejich MAC adresou a portem switche, na který jsou připojeny. Dále lze (nepovinně) každému zařízení přiřadit DNS jméno a staticky přidělit IP adresu z přiřazené VLAN sítě.

Informační systém také generuje konfiguraci pro veškeré síťové prvky, kterou na switche nahrává přes SSH. V pravidelných intervalech navíc tuto konfiguraci kontroluje a v případě zjištění možné chyby nebo změny v konfiguraci notifikuje administrátory systému. Administrátorům rovněž umožňuje prohlížet nebo upravovat konfiguraci access portů přímo přes grafické rozhraní systému.

6.1.4 Další implementovaná funkcionalita

Vzhledem k velikosti klubu Silicon Hill a množství jeho nabízených služeb byla do informačního systému integrována i správa majetku klubu a přehled čerpání financí z rozpočtu klubu.

Systém nepodporuje detailnější správu jiných členství než síťových. Pro druhy členství jsou tak vyvíjeny další systémy nebo webové stránky, pro které informační systém vystavuje rozhraní pro OAuth autentifikaci a REST API.

U členů je možné evidovat ČVUT, ISIC a další karty a informační systém je propojitelný s karetním systémem K4.

Zajímavostí je také možnost zaslat Wake on LAN packet na kterékoliv registrované zařízení.

6.1.5 Zhodnocení systému

Informační systém klubu Silicon Hill lze bez nadsázky považovat za profesionální řešení, které je plně redundantní, vertikálně škálovatelné a umožňuje spravovat a obsluhovat více než 3‘500 aktivních členství s více jak 6‘000 připojenými zařízeními včetně desítek síťových prvků a je tak skvělým možným řešením pro velké sítě, jaké má například klub Silicon Hill nebo Pod-o-lee.

Robustnost a komplexita celého řešení ale jde ruku v ruce se zvýšenou potřebou odborných znalostí a také zvýšenou náročností na údržbu celého systému, jelikož vyvíjet a spravovat plně horizontálně škálovatelné systémy není zcela triviální záležitost. To také reprezentuje fakt, že je i nyní, po více jak 5 letech, systém stále vyvíjen jeho původními tvůrci, ke kterým se od roku 2014 přidal Vladimír Kincl. Pokud by tak byl tento informační systém upraven a použit pro potřeby klubu HK, byl by s největší pravděpodobností jeho další vývoj odkázán na členy klubu Silicon Hill, nebo by se na jeho údržbu a další vývoj musela najmout externí firma.

Pokud by ale byl tento informační systém použit, znamenalo by to buď upravit návrh nové počítačové sítě klubu HK a upravit mechanizmy systému starající se o konfiguraci síťových prvků (příkazy pro konfiguraci switchů se liší výrobce od výrobce, někdy dokonce i od verze firmware), nebo upravit informační systém klubu Silicon Hill tak, aby odpovídal požadavkům této sítě, tj. schovat nebo odebrat veškerou nadbytečnou funkcionalitu a implementovat mechanizmy pro automatické přiřazování VLAN sítí. Dále by bylo potřeba doimplementovat funkcionalitu pro službu *Fitcentrum HK*.

Z těchto důvodů byl tento systém označen jako ekonomicky nevyhovující pro potřeby klubu HK, která disponuje jen cca 200 členy. Je sice pravda, že informační systém disponuje i další zajímavou funkcionalitou, jakou je např. podpora kartového systému K4, OAuth, nebo správa rozpočtu klubu a druhů členství, ale pro takto malý klub, jakým je klub HK, je tato funkcionalita nevyužitelná a o jejím zavedení se momentálně ani z dlouhodobého hlediska neuvažuje.

6.2 Informační systém klubu Pod-o-lee

Klub Pod-o-lee používal ještě do konce roku 2016 starý informační systém DUSPS implementovaný v letech 2005 až 2006 v jazyce PHP. Od roku 2016 pak začal být používán nový informační systém Hydra, který navrhl a dále vyvíjí Lukáš Kasič.

Jelikož jsou oba systémy zajímavé svým návrhem a funkcionalitou, bude níže popsána analýza obou těchto systémů a zhodnocen jejich potenciál pro možné použití pro potřeby klubu HK.

6.2.1 Informační systém DUSPS

6.2.1.1 Implementace

Frontend systému je napsán v čistém HTML, CSS a JavaScriptu, není responzivní a po grafické stránce je opravdu velmi strohý. Zajímavostí také je absence stránkování u dlouhých tabulkových výpisů (např. členů), což je údajně způsobeno omezeními použitého frameworku. Na levé straně stránky se nachází menu s výčtem dostupné funkcionality (které je opravdu požehnaně), na zbytku stránky se pak zobrazuje aktuálně používaná funkcionalita.

Backend aplikace je napsána v jazyce PHP 5.0, díky čemuž můžeme jednoznačně říci, že na dnešní poměry je aplikace již opravdu archaická. Osobně totiž hodnotím PHP jako použitelné až od verze 5.4.

Zajímavostí backendu je, že je napsán modulárně, tj. dostupnou funkcionalitu zajišťují jednotlivé moduly v aplikaci, kterých je nepřeberné množství. Další zajímavostí pak je, že téměř veškerá logika systému je implementována na úrovni PostgreSQL databáze a PHP tak slouží primárně jako prostředník pro zajištění komunikace mezi frontendem a databází a pro generování HTML kódu frontendu.

6.2.1.2 Správa členství

Kromě povinných osobních informací, které jsou definovány ve Stanovách Studentské unie ČVUT, lze v systému u člena vést i jeho rodné číslo, název školy a dokonce fakultu, kterou studuje.

Systém rovněž umožňuje vést i evidenci druhů členství členů (*síťové členství, členství ve fitcentru* ad.), u kterých lze zároveň definovat i výši členského příspěvku a variabilní symbol.

Samozřejmostí je pak automatická aktivace nebo prodloužení členství při přijetí příspěvku a jeho deaktivace po vypršení jeho doby splatnosti. Zde je zajímavé, že informace o přijatých zprávách se získávají z emailů zasílaných bankou. Toto řešení se ale postupem času ukázalo jako nevhodné, protože banka pravidelně mění strukturu těchto emailů, v důsledku čehož je pak vždy nutné upravit procedury, které tyto emaily zpracovávají a extrahují z nich potřebné informace.

Dále systém umožňuje velmi precizní nastavení oprávnění. Téměř ke každé funkci v systému lze udělit nebo omezit přístup a přes grafické rozhraní lze definovat i nové uživatelské role a skupiny.

Celkově je ale správa velmi nepřehledná, neexistuje totiž nějaké sjednocující uživatelské rozhraní pro nastavení konkrétního člena (např. v podobě karet, modálních oken atd.). Pokud je tedy potřeba u člena upravit více údajů, např. osobní informace a připojená zařízení, musí se administrátor přepnout nejprve do funkcionality (modulu) pro správu osobních informací, zde si člena vyhledat a provést změny a následně se přepnout do funkcionality (modulu) sítě, znovu člena (resp. jeho zařízení) vyhledat a provést potřebné úpravy. Nepřehlednost systému pak podtrhává i již dříve zmíněná absence stránkování.

6.2.1.3 Správa počítačové sítě

Na Podolské koleji jsou zařízení všech uživatelů vázána MAC adresou zařízení na zásuvky v pokoji, na kterém bydlí. Infomační systém tak nabízí správu MAC adres zařízení s přiřazeným pokojem. Dále je možné v systému evidovat síťové prvky (switche), jejich porty, bloky koleje, pokoje na blocích, zásuvky na pokojích s mapováním na konkrétní port switche a rozsahy dostupných IP adres z veřejného a neveřejného rozsahu.

Každému uživateli (členovi) je pak přiřazena jediná IP adresa, v důsledku čehož může v danou chvíli používat pouze jedno zařízení. Toto omezení se ale nevztahuje na Wi-Fi, kde je využíván logovaný NAT.

Zabezpečení sítě pak zajišťuje MAC port-security na portech switche (pokojových zásuvkách) a statický ARP[24].

Systém dále umožňuje zobrazovat logy ze síťových prvků, zobrazovat informace o konfiguraci a stavu síťových prvků přes SNMP[25], evidovat servery, spravovat DNS záznamy a mnoho dalšího. Většina této funkcionality je ale dnes již nefunkční v důsledku změny sítě, pro kterou byla tato funkcionalita navržena.

6.2.1.4 Další implementovaná funkcionalita

Mimo již uvedené umí informační systém zobrazit i detailní logy. Logována je přitom téměř veškerá aktivita, tj. kdo, kdy a co v systému upravoval, jaký byl stav před a po provedení změny, a dokonce i kdo k jakým informacím přistupoval, a to včetně osobních informací a nastavení členů.

I když je tento systém logování velmi propracovaný, tak v důsledku změn při dalším vývoji je v současné době rovněž nefunkční.

Jako poslední pak stojí rovněž za zmínku napojení na kartový systém K4, který je na Podolí téměř všude instalován, a tedy i evidence karet jednotlivých členů.

6.2.1.5 Zhodnocení systému

Informační systém sice nabízí bohatou funkcionalitu, nicméně velká část z ní již nefunguje a bylo by tak potřeba buďto tuto funkcionalitu opravit, nebo ji ze systému odstranit. Členění funkcionality do značného množství jednotlivých modulů a celková obsáhlost dostupné funkcionality se postupem času ukázala jako kontraproduktivní, systém tak bylo celkově náročné dále vyvíjet, nebylo možné dostatečně rychle reagovat na měnící se požadavky, změny v API napojených služeb a HW. Nakonec tak ze systému zbylo funkční pouze jádro použitého frameworku, které se ale podle slov L. Kasiče nedá moc upravovat.

Jako zásadní nedostatek (problém) pak hodnotím přenesenou logiku aplikace do PostgreSQL databáze. Ohledně tohoto způsobu návrhu aplikací se vede řada diskuzí, přičemž komunita je vesměs rozdělena na dva tábory, odpůrce a zastánce. Oba způsoby návrhu (mít logiku v aplikaci, nebo v databázi) totiž mají své výhody i nevýhody. Osobně jsem toho názoru, že logika by se měla držet v aplikaci a ne databázi, která primárně slouží k ukládání dat a provádění výpočtů nad uloženými daty, nicméně mohou existovat případy, kdy vytvoření procedur nad databází dává smysl, např. při potřebě zajistit, že daná operace bude provedena atomicky, v přesně dané posloupnosti a že během zpracování nebude vyhodnocen jiný příkaz, který může výpočet negativně ovlivnit (např. že během výpočtu zůstatku na účtu neproběhne další transakce). Tyto případy lze sice zabezpečit i na straně aplikace za pomocí zvýšení stupně izolace, transakcí a uzamykání záznamů či celých tabulek, nicméně zajištění 100 % spolehlivosti je obtížné a snadno se přehlédne nějaký neošetřený stav. Navíc systém zamykání tabulek má negativní vliv na celkový výkon databáze.

Dalším problémem při potenciálním použití tohoto systému je, že již neexistují zdrojové kódy procedur napsané v jazyce C, v důsledku čehož by pak musela být celá logika aplikace napsána znovu nebo zpětně zjištěna pomocí reverzního inženýrství.

Problémem je rovněž napojení na LDAP databázi, které je opět realizováno přes procedury v databázi. Při výpadku / odstávce LDAP databáze tak totiž neexistuje možnost zpětné synchronizace změn.

---

[24] ARP – Address Resolution Protocol, služba pro zjištění MAC adresy zařízení v místní síti na základě známé IP adresy.

[25] SNMP – Simple Network Management Protocol, protokol sloužící ke správě sítí a síťových zařízení.

Vzhledem k těmto zjištěním proto hodnotím systém DUSPS jako nevyhovující a nebude tak uvažován ani jako podklad pro systém nový.

6.2.2 Informační systém Hydra

Vzhledem k nedostatkům původního řešení, které je popsáno v kapitole 6.2.1, a absenci zdrojových kódů PostgreSQL procedur, vznikla potřeba nasadit systém jiný.

Lukáš Kasič proto navrhl nový informační systém Hydra, který začal v roce 2016 implementovat. Byť je systém stále ještě ve fázi intenzivního vývoje a celkově byl již několikrát kompletně přepsán, stojí za úvahu případná participace na vývoji tohoto systému s cílem nasadit výsledný systém nejen na Podolské koleji, ale rovněž na Hlávkově koleji.

Samotný systém je pak zajímavý i z hlediska jeho návrhu, protože funkční celek tvoří jednotlivé mikro-služby, z čehož je pak odvozen i jeho název Hydra (vícehlavá příšera).

6.2.2.1 Implementace

Systém se skládá z jednotlivých mikro-služeb, které jsou napsány v jazyce Python. Jednotlivé části celého systému pak v současnosti tvoří tyto mikro-služby:

- rozhraní pro základní přehled a správu uživatelů;
- administrace celého systému;
- mikro-služba pro komunikaci s bankovním systémem;
- mikro-služba pro kartový systém K4.

Běžní uživatelé přitom mají přístup pouze do první služby, ostatní služby pak spravuje přímo administrátor. Veškeré služby jsou ovládány přes webové rozhraní (frontend), jejichž uživatelské rozhraní je implementováno ve Twitter Bootstrap frameworku, případně jiné responzivní knihovně.

Služby jsou implementovány hlavně v jazyku Python a frameworku Django [8], přičemž je využito automatické tvorby administrace podle definovaného datového modelu, díky čemuž pak výsledné rozhraní mikro-služby více méně opisuje databázový model služby a v konečném výsledku tak působí jako zdokonalené grafické rozhraní databáze, které je upraveno pro konkrétní potřeby napojené aplikace.

Komunikaci mezi jednotlivými mikro-službami pak zajišťuje RabbitMQ [9], tj. systém pro zasílání zpráv. Celý systém by navíc měl být plně horizontálně škálovatelný, čímž odpadá problém redundance a load balancingu.

6.2.2.2 Správa členství

Rozhraní v základu umožňuje pouze registraci a základní správu členů, tj. evidenci osobních informací (včetně pokoje, na kterém člen bydlí a fakultu, na které studuje), síťových zařízení, karet K4 a zobrazení informací o dostupných službách, použití karty, platbách a historii užívaných služeb.

Je-li potřeba měnit jiné nastavení, přiřazovat systémové role, skupiny či cokoliv jiného (IP rozsahy, přístupové údaje, seznam VLAN, platby, odpustky, tresty, tj. BANy, bloky, pokoje atd.) je potřeba dělat tyto úpravy v administraci, tj. přes rozhraní jiné mikro-služby.

6.2.2.3 Správa počítačové sítě

Správa sítě kopíruje původní systém, tj. vede se evidence bloků, pokojů, zásuvek, switchů a jejich portů. Na metalické síti je aplikována MAC port-security a systém rovněž generuje statickou ARP tabulku pro směrování. U každého zařízení se pak vede typ (notebook, mobil), podle čehož pak systém nabízí počet MAC adres, které je možné u tohoto zařízení evidovat. Dále se vede evidence již zmíněných VLAN a k nim přiřazených IP rozsahů, DNS záznamy a mnoho dalšího.

Propagace nové konfigurace se pak provádí v 15-ti minutových intervalech, přičemž i tento systém má být do budoucna napojen na systém RabbitMQ, čímž se dosáhne rychlejší propagace nové konfigurace, která se navíc bude generovat jen v případě, že došlo k nějaké změně.

6.2.2.4 Další implementovaná funkcionalita

Kromě již zmíněné integrace karetního systému K4 umožňuje informační systém vést i evidenci ceníků služeb, které navíc lze naplánovat i do budoucna. Zajímavostí jsou rovněž tzv. promo služby, kdy je členovi, zpravidla po dobu několika dnů, umožněno využívat některou ze služeb, kterou nemá zaplacenou.

Do budoucna se pak očekává implementace a napojení dalších systémů pomocí mikro-služeb, stejně jako u předchozího systému, tj. SNMP, logovací systém ad.

6.2.2.5 Zhodnocení systému

Systém je velmi zajímavý svým návrhem, jenž je koncipován pomocí mikro-služeb, ve kterých osobně vidím velký potenciál. Aplikaci jako celek tvoří jednotlivé služby, jež by měly být pokud možno co nejvíce minimalistické a všestranné. Jednotlivé služby pak mohou být vyvíjeny nezávisle různými programátorskými týmy, jejich vývoj tak může být rychlejší, lépe se testují a výpadek nebo odstávka jedné služby nemusí ovlivnit systém jakožto funkční celek.

I přes tyto a další klady si ale nemyslím, že je koncept mikro-služeb vhodný pro toto řešení. Důvodem je, že kolejní informační systémy jsou zpravidla vyvíjeny jednotlivci, díky čemuž odpadá jedna z hlavních výhod tohoto návrhu, tj. rozdělení vývoje mezi více vývojářů. Další nevýhodou je pak podstatně větší čas potřebný k návrhu a implementaci celého systému, jelikož je potřeba navrhnout a implementovat rozhraní, přes která budou služby mezi sebou komunikovat, tj. služba musí vystavovat API (přijímat a zasílat zprávy) a systémy (služby), které jsou na tuto službu napojeny, pak musí umět s tímto API komunikovat.

Další vlastností, se kterou nejsem úplně ztotožněn je, že systém jakožto celek nelze ovládat (spravovat) přes jednotné (zastřešující) rozhraní a každá mikro-služba vystavuje vlastní grafické rozhraní (administraci) pro svou správu. Jako nevhodný hodnotím rovněž návrh administrace těchto mikro-služeb, který opisuje databázovou strukturu, v důsledku čehož nelze službu ovládat bez znalosti datového modelu, který používá. Správce, který chce provést v systému nějakou úpravu, tak musí vědět, kde jakou entitu musí vytvořit / smazat / upravit, jaké hodnoty ji má nastavit a jak jsou entity mezi sebou provázány.

Jako nevhodné pro potřeby klubu HK rovněž vidím to, že je systém horizontálně škálovatelný, což klade další nároky na potřebnou odbornou znalost, tj. stejný problém jako u informačního systému Silicon Hill. Systémy pro menší koleje by měly být, podle mého názoru, pokud možno co nejjednodušší z hlediska návrhu a odborných znalostí potřebných k jejich dalšímu vývoji. Díky tomu pak bude větší šance najít dalšího dobrovolníka, který bude mít dostatečné odborné znalosti pro jejich další vývoj.

Horizontálně škálovatelné systémy a návrh v podobě mikro-služeb tak mají význam hlavně u větších kolejí (Podolí, Strahov) nebo v případě, kdy by se uvažovalo o vytvoření centrálního informačního systému společného pro všechny koleje ČVUT.

Z těchto důvodu hodnotím systém Hydra jako nevyhovující pro potřeby klubu HK.

6.3 Informační systém klubu Masařka

Studentský klub Masařka (Masarykova kolej) používá od roku 2008 informační systém DIS, též přezdívaný Dezinformační systém, což už ze samotného názvu nevěstí moc dobrého. Systém je totiž velmi nepřehledný, nemá responzivní design a v některých případech dokonce umožňuje i chybnou konfiguraci síťových zařízení.

6.3.1 Implementace

DIS je napsán za použití čistého HTML, CSS a PHP 5.2, tj. bez použití knihoven nebo frameworků, až na výjimku šablonovacího PHP frameworku Smarty [10], který umožňuje snazší generování HTML kódu. Jako databázi používá PostgreSQL.

Součástí informačního systému je rovněž řada podpůrných skriptů napsaných v jazycích PHP, Bash, Python, Ruby a dalších, starajících se o automatizaci některých procesů, stahování informací o platbách a generování konfigurace pro síťové prvky. Zajímavostí také je, že vygenerování a nasazení nové konfigurace pro síťové prvky probíhá z části manuálně. Po změně nastavení access portů musí administrátor systému manuálně vynutit vygenerování a nasazení nové konfigurace, k čemuž v systému slouží tlačítko *Commit*. Součástí systému je pak i log s informacemi, kdo kdy provedl poslední commit.

Na otázku ohledně bezpečnosti celého systému mi bohužel nebyli schopni stávající správci systému s jistotou odpovědět. Dle jejich slov by měl být systém odolný minimálně vůči SQL Injection a XSS, nicméně stoprocentně si jisti nejsou.

6.3.2 Evidence členů a členství

DIS umožňuje vést všechny nezbytné osobní informace o členech klubu (viz kapitola 4.2). Zajímavostí je, že systém nepodporuje uživatelské role, resp. rozlišuje pouze, jestli má uživatel přístup do administrace systému či nikoliv.

Další zajímavostí pak je, že zúčtovací období (semestry) se musejí do systému zavádět ručně, což v případě opomenutí může vést až k deaktivaci členství všech členů a jejich odpojení od počítačové sítě.

6.3.3 Správa počítačové sítě

U každého člena se vede seznam jeho zařízení společně s MAC adresou, přiděleným switch portem a IP adresou. Zde je mimo jiné systém náchylný na špatnou konfiguraci. Důvodem je, že metalická a bezdrátová síť používají rozdílné IP rozsahy a administrátor tak musí zařízení přiřadit IP adresu ze správného rozsahu, jinak se uživatel nebude schopen připojit.

Systém také generuje statickou ARP konfiguraci pro hlavní router, tj. zařízení musí mít nastavenou IP adresu, kterou mají přiřazenou v systému, jinak nebude schopné v síti komunikovat. Dále se generuje konfigurace pro DHCP a patrové switche. RADIUS server získává informace o zařízeních přímo z PostgreSQL databáze a je použit jak pro bezdrátovou, tak pro metalickou síť, kde kontroluje, zda se na port switche připojuje zařízení s MAC adresou, které je na tomto portu registrována v DISu.

I když je tedy RAIDUS server napojen přímo na databázi systému, musí uživatel počítačové sítě po změně konfigurace (např. z důvodu přestěhování na jiný pokoj) vyčkat až 15 minut na projevení změn. Důvodem je to, že konfigurace pro router a DHCP je generována v 15ti minutových intervalech a promítnutí těchto změn na síťové prvky tak není okamžité.

Zásadním nedostatkem tohoto řešení ale je, že neeviduje historii přiřazených IP adres, v důsledku čehož nemusí být bývalý uživatel IP zpětně dohledán.

6.3.4 Další implementovaná funkcionalita

Kromě již uvedené funkcionality umožňuje systém vést i inventarizaci síťového HW. Tato funkcionalita ale prakticky nikdy nebyla využívána.

Další zajímavostí je implementovaný tzv. kyblíkový systém. Jedná se o archaický pozůstatek z dob, kdy měli uživatelé počítačové sítě nastavený FUP[26] limit, který byl pro snazší pochopení reprezentován kyblíky, kdy po naplnění všech kyblíků byla uživateli omezena rychlost připojení.

6.3.5 Zhodnocení systému

Informační systém DIS je velmi zastaralý a neodpovídá dnes již běžným standardům. Pokud by měl být použit pro potřeby klubu HK, musel by být minimálně přepsán celý frontend systému a drtivá většina backendu. Z těchto důvodů se jeví jako vhodnější řešení použití jiného systému, nebo implementace systému zcela nového. Toto tvrzení podporuje i fakt, že už delší dobu se na Masařce

[26] FUP – Fair User Policy, politika omezující maximální množství přenesených dat.

uvažuje o implementaci zcela nového systému, který odstraní nedostatky toho stávajícího a více zjednoduší a zpřehlední celou správu členů a sítě.

6.4 Informační systém klubu Sincoolka

Studentský klub Sincoolka působí na Sinkuleho a Dejvické koleji. Pro evidenci členů a správu přístupů pro počítačovou síť a fitcentrum používá od roku 2010 informační systém SINIS.

6.4.1 Implementace

SINIS je implementován za použití čistého HTML, CSS a PHP 5.2. Jak bylo zjištěno, potýká se systém s řadou funkcionálních i bezpečnostních chyb a na mnohých místech je zranitelný i těmi nejběžnějšími druhy útoků, jako je například SQL Injection.

Jako databáze byla vybrána PostgreSQL, na kterou je napojen informační systém společně s řadou pomocných skriptů, napsaných v nejrůznějších programovacích jazycích, které generují konfigurace pro statický ARP, DHCP, DNS a další síťové prvky, stahují platby atd. Ke generování a nasazení nové konfigurace dochází každých 15 minut.

Jedná se o velmi jednoduchý systém, který se implementačně velmi blíží k DISu (informační systém klubu Masařka). Frontend systému sice také není responzivní, nicméně je mnohem přehlednější a intuitivnější na ovládání.

6.4.2 Evidence členů a členství

Stejně jako DIS umožnuje i SINIS uchovávat o členech základní osobní informace společně s číslem OP, studovanou univerzitou, kolejí a pokojem, na kterém člen bydlí.

Zajímavostí je navíc způsob řízení přístupu. Systém sice nemá vyloženě definované uživatelské role, nicméně uživatelům lze nastavovat oprávnění, tj. povolovat nebo zakazovat konkrétní funkcionalitu. Přes toto řízení přístupu se nastavuje i typ členství, tj. *základní, síťové, fitcentrum* apod. Navíc systém umožňuje na kartě člena (uživatele) zobrazit historii přidělených oprávnění.

6.4.3 Správa počítačové sítě

U každého člena lze držet seznam připojených zařízení společně s jejich MAC adresou a přirazenou IP adresou a přiřazeným portem na patrovém switchi. Při registraci nového zařízení systém automaticky přiřadí tomuto zařízení volnou IP adresu. Navíc systém umožňuje vést kompletní historii všech přiřazených IP adres.

Na metalické síti je uživatel vázán ke konkrétnímu portu patrového switche podle nastavené MAC port-security. Na bezdrátové síti se uživatel ověřuje svými přístupovými údaji přes RADIUS server.

6.4.4 Další implementovaná funkcionalita

Systém umožňuje u členů evidovat karty ISIC, ČVUT a další. Pro tyto účely si dokonce studenti ze Sinkuleho a Dejvické koleje vytvořili vlastní kapesní verzi čtečky karet, která je se systémem SINIS propojena přes Wi-Fi. Pomocí evidovaných karet si pak člen může jednoduše otevřít dveře do fitcentra a jiných místností, které má v systému povoleny.

Systém umožňuje evidovat členy fitcentra, což je také dáno propojením se systémem bezkontaktních karet.

Další zajímavostí je, že systém umožňuje vygenerovat QR kód pro snadnou úhradu členského příspěvku a obsahuje implementovaný volební systém pro volby do představenstva klubu.

6.4.5 Zhodnocení systému

Informační systém SINIS je velmi zastaralý a neodpovídá dnes již běžným standardům. Pokud by měl být použit pro potřeby klubu HK, musel by být s největší pravděpodobností zcela přepsán, což je dílem dáno četností a závažností bezpečnostních děr. Z těchto důvodů se jeví jako vhodnější řešení

použití jiného systému nebo implementace systému zcela nového, byť systém implementuje poměrně zajímavou funkcionalitu (QR platby, volby).

6.5 Informační systém klubu Buben

Správci informačního systému KrlStin, který se používá na Bubenečské koleji a na koleji v Kladně, bohužel odmítli participaci v rámci této diplomové práce. Podařilo se však zjistit, že informační systém by měl být napsán v jazyce Ruby on Rails a grafické rozhraní vytvořené pomocí knihovny Bootstrap.

Informační systém je cca z roku 2001, tedy zhruba 16 let starý a v současnosti se obě koleje částečně potýkají s problémem dalšího vývoje tohoto systému z důvodu nedostatku programátorů.

Vzhledem k absenci dalších informací a odmítnutí participace a stáří informačního systému nebude o tomto informačním systému dále uvažováno pro účely této práce.

6.6 Informační systém klubu Orlík

V době psaní této práce klub Orlík nedisponuje informačním systémem pro správu evidence členů a řízení jejich přístupu k službám klubu.

Původně klub informačním systémem disponoval, nicméně v důsledku změn infrastruktury počítačové sítě a následné změny struktury databáze přestalo fungovat webové grafické rozhraní a přidružené mechanizmy. Veškeré změny v nastavení a evidenci členů tak musejí být prováděny manuálně přímo v MySQL databázi, což pochopitelně občas vede k různým chybám a opomenutím.

V době psaní této práce tak klub Orlík hledá programátory či jiný informační systém, který by tento nedostatek odstranil, jelikož současný stav je z dlouhodobého hlediska neudržitelný.

6.7 Zhodnocení

Analýzou existujících řešení bylo zjištěno, že pro potřeby klubu Hlávkova kolej nelze doporučit žádný z dostupných informačních systémů. Jednotlivé systémy jsou buď příliš zastaralé a některé kolejní kluby tak dokonce uvažují o jejich aktualizaci či nahrazení jiným systémem, nebo jsou moc robustní a rozsáhlé a vyžadují tak mnohem větší odborné znalosti jednak pro správu (údržbu) a jednak pro další případný vývoj.

Možnou alternativou by tak mohlo být využití některého ze stávajících systémů jako podklad pro systém nový, který by tyto nedostatky a problémy odstranil, nicméně z důvodu neznalosti implementace těchto systémů a rozsahu potřebných změn by takováto úprava existujícího systému trvala přinejmenším stejně dlouho, jako implementace systému nového. Z tohoto důvodu se tato varianta nejeví jako vhodná a ekonomická.

V následující kapitole proto bude představen návrh nového systému, který bude následně implementován a otestován.

I když nebude použito žádné z již existujících řešení, byla i tak analýza těchto řešení, a především osobní schůze a konzultace s jejich vývojáři, značným přínosem. Každý z informačních systémů byl navržen za použití jiných technologií a osobního přístupu vývojáře, díky čemuž lze nyní při návrhu nového systému zohlednit zjištěná fakta, výhody a nevýhody jednotlivých přístupů a rady vývojářů.

Jelikož má klub Orlík téměř totožné požadavky na systém jako klub Hlávkova kolej, bude výsledný systém nabídnut klubu Orlík k použití.

7. Návrh nového systému

Při analýze existujících řešení bylo zjištěno, že většina existujících řešení (informačních systémů) je značně zastaralá a většina rovněž nebyla od doby svého uvedení do produkce dále vyvíjena a rozšiřována, což způsobilo postupný úpadek a ztrátu funkčnosti některých systémů.

Nový informační systém proto bude navrhnut především s ohledem na dnešní běžně používané trendy, přičemž hlavní důraz bude kladen na jednoduchost systému a hlavně na jeho snadnou rozšiřitelnost.

7.1 Použité technologie a knihovny

Výběr použitých technologií byl řízen hlavně podle dnešních běžných trendů, kvalitní dokumentaci knihoven a velké komunitě, která tyto technologie používá, díky čemuž by pak měla být větší šance, že se dalšího vývoje systému ujme jiný programátor. Důraz byl také kladen na využití Open-Source knihoven a frameworků, které jsou v současné době intenzivně vyvíjeny a udržovány, a u kterých lze předpokládat, že budou vyvíjeny i dále v budoucnosti.

Pro frontend proto bude použit HTML, CSS a JS knihovna založená na Twitter Bootstap [7], což je v současné době nejpoužívanější Open-Source HTML, CSS a JS knihovna, která je známá především pro svůj responzivní design, výbornou dokumentaci a bohatou paletu předpřipravených stylů a UI[27] komponent (formuláře, tabulky, modální okna a mnoho dalšího).

Pro backend aplikace pak byl zvolen jazyk PHP 7.0 (či vyšší) a Nette Framework [11]. Důvodem pro zvolení PHP je předpoklad, že informační systém bude dále vyvíjen hlavně správci sítě, přičemž téměř každý správce sítě se většinou dříve či později s PHP setká, protože jde o nejpoužívanější jazyk pro webové prezentace současnosti. Je sice pravda, že PHP je v současné době na úpadku a novodobým trendem jsou hlavně jazyky Python nebo Ruby on Rails, nicméně tyto jazyky neovládám a mohl bych tak při pokusu o implementaci chybně navrhnout strukturu aplikace a zavést nevhodné best-practices. S PHP mám naopak více jak osmiletou praxi a s Nette Frameworkem pracuji déle než 4 roky, díky čemuž jsem si jist výsledným kvalitním návrhem a implementací aplikace. Pro účely této práce tedy bude upřednostněn jazyk PHP především pro jeho dokonalou znalost.

Nette Framework byl vybrán hlavně pro jeho jednoduchost, velmi kvalitní zabezpečení proti různým druhům útoků (XSS, CSRF, session hijacking, session fixation ad.), rychlost a také jeho dokonalou znalost (Nette Framework dokonale znám a několikrát jsem do tohoto projektu přispíval) a skvělou českou komunitu (jelikož se jedná o český projekt).

Dále budou v krátkosti představeny některé z hlavních knihoven vybraných při návrhu pro implementaci výsledného systému.

7.1.1 Frontend

7.1.1.1 AdminLTE

AdminLTE [12] je Open-Source knihovna založená na Twitter Bootstrap navržená hlavně pro použití pro administrátorské rozhraní aplikací nebo jako UI pro různé systémy. Tato knihovna je velmi populární, aktivně vyvíjená a disponuje skvělou dokumentací a demoverzí, která předvádí veškerou implementovanou funkcionalitu. Ve výsledku tak při implementaci vlastní aplikace stačí pouze zkopírovat patřičný HTML kód z demoverze a upravit jeho obsah s téměř minimální potřebou zasahovat do CSS a JS kódu knihovny. Díky tomu je vývoj aplikace rapidně urychlen a vývojář (programátor) se tak hlavně soustředí na implementaci funkcionality než grafického rozhraní (které má již z většiny díky této knihovně předpřipravené).

Další předností této knihovny je pak responzivní design, skvělá podpora v moderních i starších prohlížečích a nativní podpora a integrace velkého množství nejpoužívanějších JS a HTML knihoven,

[27] UI – User interface, tj. uživatelské rozhraní.

jako je například *Date Picker, Color Picker, Time Picker, iCheck, Input Mask, Select2, WYSIHTML5, DataTables, jQuery* a mnoho dalšího. Dále si v krátkosti představíme některé z nejzajímavějších knihoven, které je plánováno použít pro výslednou implementaci.

7.1.1.2 jQuery

jQuery [13] je JS knihovna navržená pro zjednodušení implementace JS kódu klientské části aplikace. Jedná se o bezplatný Open-Source software, který se těší velké popularitě a na kterém je postavena velká část moderních webových stránek.

jQuery syntaxe je navržená pro snazší orientaci v HTML dokumentu, selekci HTML elementů, tvorbu animací, zachytávání událostí a vývoji AJAX aplikací. jQuery rovněž nabízí možnost implementace vlastních rozšíření, díky čemuž vznikl nespočet JS knihoven rozšiřujících jQuery o další zajímavou funkcionalitu (např. většina knihoven, které integruje šablona AdminLTE).

jQuery bude použita jednak z důvodu, že tvoří jádro Twitter Bootstrapu (a tedy i AdminLTE), ale také pro implementaci vlastního obslužného kódu, především pro pohodlný výběr HTML elementů a zachytávání událostí (např. klik na tlačítko)

7.1.1.3 DataTables

Vzhledem k tomu, že původní verze aplikace obsahuje přes 1‘000 uživatelů (členů) a tento počet bude, stejně jako další data (např. bankovní transakce), dále narůstat, bude potřeba tato data přehledně interpretovat (zobrazovat) v aplikaci. Jednou z možností je např. stránkovaný výpis s implementovaným formulářem pro vyhledávání záznamů. Implementace takových přehledů je ale pracná a neintuitivní.

Proto bude využita knihovna DataTables [14], která umožňuje jednoduše vykreslit tabulky obsahující i 100‘000 záznamů, implementuje fulltextové vyhledávání, řazení podle sloupců, stránkovaný výpis a mnoho dalšího. Další předností této knihovny je integrace AJAX-ového načítání dat (tj. načítání záznamů na pozadí klientské části aplikace) a možnost přesunutí zpracování dat na server, kdy se vyhledávání a řazení záznamů provádí na serveru (ideálně na úrovni databáze) a klientovi pak nejsou zasílány všechna data, ale pouze data ze stránky, kterou si zrovna prohlíží.

Knihovnu lze použít samostatně, nebo jako rozšíření pro jQuery.

7.1.1.4 Nette JS

Jedná se o minimalistickou knihovnu, která byla kdysi součástí Nette Sandboxu (šablony pro nový Nette projekt). V repositářích na Githubu se zachovala pouze její část řešící obsluhu validace Nette formulářů, nicméně knihovna se mi především kvůli její jednoduchosti a integraci s funkcionalitou Nette Frameworku zalíbila natolik, že jsem si ji lehce rozšířil a používám ji u všech projektů, které implementuji v Nette Frameworku.

Knihovna v základu řešila pouze zpracování odpovědi Nette Frameworku zasílané pro AJAX odkazy a formuláře a prováděla validaci Nette formulářů, kdy při zjištění nevalidního formuláře zrušila jeho odeslání a zobrazila výstražné dialogové okno s odůvodněním.

V průběhu let práce s Nette Frameworkem jsem tuto knihovnu rozšířil o další funkcionalitu, mezi tu nejzajímavější pak patří tato:

- Pokud má odkaz nebo formulář nastavený styl *ajax*, pak se jeho zpracování provádí AJAX-em na pozadí klientské části aplikace (tj. bez potřeby překreslení celé stránky).
- Veškeré AJAX dotazy se ukládají do historie prohlížeče, díky čemuž lze procházet historii zpět a dopředu (provádět dotazy opakovaně). Toto chování lze zrušit přidáním stylu *dont-track.*
- Původně probíhala validace formulářů postupně a vypsala se pouze první chybová zpráva, na kterou validace narazila, tj. pokud bylo ve formuláři porušeno více validačních pravidel, byl uživatel informován pouze o prvním z nich, vstup opravil, formulář zkusil odeslat znovu a až následně byl informován o porušení dalšího validačního pravidla. Toto chování nebylo uživatelsky přívětivé, a proto jsem ho upravil tak, aby se vypisovala všechna porušená

validační pravidla najednou. Navíc byla přidána možnost vlastní obsluhy chybových hlášek, které tak lze uživateli zobrazovat i jinak než v podobě varovného dialogového okna.
- Jako poslední pak byla přidána podpora pro Twitter Bootstrap formuláře, kdy se na formulářové prvky, které neprošly validačními pravidly, aplikují Bootstrap styly, které tyto prvky zvýrazní (v závislosti na použité Bootstrap knihovně, většinou jim zčervenají okraje).

7.1.1.5 Input Mask

Input Mask [15] je JS knihovna, která upravuje textové vstupy formulářů a zajišťuje, že data budou zadány uživatelem v předem definovaném formátu, typicky pak, že datum bude zadáno například ve formátu *DD/MM/YY*, telefonní číslo ve formátu *(+420) XXX XXX XXX* atd. Navíc uživateli při zadávání zobrazuje masku požadovaného vstupu, díky čemuž je pro uživatele vyplňování formuláře více přívětivé.

V základu umí tato knihovna zobrazovat masku pro datum a čas (různých formátů), čísla, telefonní čísla, email, IP adresu, MAC adresu, URL adresu a mnoho dalšího. V případě potřeby je rovněž možné definovat vlastní masky vstupů, a to i na základě pokročilých regulárních výrazů.

Knihovna je Open-Source, ke své funkci vyžaduje jQuery a bude využita především pro zadávání správného formátu MAC adres, data, času a telefonního čísla, tj. pro vstupy, které lze standardně zapsat v různých formátech (v závislosti na systému, národnosti a zvyklostem).

7.1.1.6 Are You Sure

Vzhledem k tomu, že návrh uživatelské části aplikace počítá se zpracováním příkazů na pozadí pomoci AJAXu, pomocí kterého bude rovněž dynamicky překreslován obsah grafického rozhraní, je potřeba zajistit, že uživatel (administrátor) před odchodem ze stránky (uzavřením prohlížeče) změněný obsah formulářů uloží.

Z těchto důvodů bude použito další minimalistické Open-Source rozšíření pro jQuery s názvem Are You Sure [16], které při pokusu o odchod ze stránky nebo uzavření okna prohlížeče kontroluje, zda nebyly změněny obsahy formulářů a v případě zjištění neuložené změny zobrazí dialogové okno s varováním. Uživatel tak může akci zrušit a změny ve formuláři před odchodem ze stránky uložit.

Navíc rozšíření zakáže odeslání formuláře a deaktivuje tlačítko pro odeslání v případě, kdy nebyly ve formuláři provedeny změny, díky čemuž se efektivně zabrání vykonání zbytečného dotazu na server.

7.1.1.7 Toastr

Při provádění různých akcí (odeslání formuláře, klik na tlačítko ad.) bude potřeba uživatele informovat o vykonání (případně důvodu zamítnutí) akce (např. formulář neprošel validací, záznam již neexistuje atd.).

Pro tyto účely byla zvolena knihovna Toastr [17], která umožňuje zobrazovat přehledné a neblokující vyskakující notifikace v rohu okna prohlížeče. Toastr umožňuje v základu zobrazovat hned několik druhů typů notifikací, a to informativní (modré), varovné (žluté), chybové (červené) a úspěšné (zelené), přičemž vzhled notifikací lze samozřejmě nastylovat podle potřeby.

U notifikací je možné nastavit titulek, zprávu, typ, umístění, efekt pro zobrazení a skrytí, automatické skrytí po uplynutí definovaného časového úseku a mnoho dalšího.

7.1.2 Backend

7.1.2.1 Nette Framework

Hlavní důvody pro vybrání tohoto frameworku byly jeho jednoduchost (strmá křivka učení), velmi kvalitní zabezpečení proti různým druhům útoků (XSS, CSRF, session hijacking, session fixation ad.), rychlost a také jeho dokonalou znalost.

Jedná se o český Open-Source projekt, který využívá MVP[28] architekturu, nabízí širokou paletu základní funkcionality, skvělou implementaci UI komponent, především pak formulářových prvků. Novější verze Nette vyžadují PHP 5.6, ideálně pak 7.0 a vyšší.

Řada vývojářů Nette kritizuje hlavně kvůli jeho přílišné jednoduchosti (framework totiž nativně neobsahuje autentifikační ad. mechanizmy nebo překladač, pro který je definován pouze interface, jehož obsluhu si už musí programátor doimplementovat sám). Postupem času ale vzniklo nepřeberné množství knihoven a rozšíření pro Nette, díky čemuž byly tyto nedostatky odstraněny. Samotný Nette Framework byl navíc během času rozdělen do několika menších knihoven, a programátor si tak může složit projekt pouze z knihoven, které pro výsledný produkt potřebuje, díky čemuž je pak výsledná aplikace menší a rychlejší.

Mezi další přednosti Nette pak patří skvělý konfigurační, emailový, ladící a reportovací systém. Ve vývojovém prostředí je pak uživateli zobrazen ladící panel, kde se zobrazují informace o zpracování dotazů (zátěž na CPU, použitá paměť, doba zpracování požadavku), přesměrování, ke kterým došlo, a informace o přihlášeném uživateli. Jednotlivá rozšíření pro Nette se pak mohou do tohoto panelu integrovat a zobrazovat tak další informace například o chybějících jazykových překladech, dotazech na databázi a seznam událostí, ke kterým během zpracování došlo spolu s informací o jejich zpracování. Reportovací systém pak přehledně hlásí veškeré chyby, ke kterým během zpracování došlo spolu s detailní trasou volaných funkcí, jejich argumenty, úryvky kódu, kde k chybě došlo a stavu systému. V produkčním prostředí jsou pak tyto chyby zapisovány do HTML souborů a vývojáři je obratem zaslána emailová notifikace obsahující místo v kódu, kde k chybě došlo, na jaké URL adrese a popis chyby. Díky tomu lze pohotově reagovat na případné problémy, snadno detekovat jejich příčinu a vydávat aktualizace, které chyby odstraní.

7.1.2.2 Kdyby/Translation

Kdyby/Translation [18] je knihovna pro Nette Framework z projektu Kdyby založená na Symfony [19]. Nabízí bohatou paletu funkcionality, především pak možnost detekce jazyka uživatele na základě několika zdrojů (informace z prohlížeče, session ad.) a integrace do Nette ladícího panelu, šablonového systému a UI komponent.

Překlady jsou načítány dynamicky z *NEON* souborů (formát sloužící především pro konfiguraci Nette a je velmi podobný formátu *YAML*), lze je zapisovat do hierarchické struktury a samozřejmostí je i podpora skloňování.

7.1.2.3 Kdyby/Console

Kdyby/Console [20] je knihovna pro Nette Framework z projektu Kdyby založená na Symfony. Knihovna umožňuje implementovat do výsledné aplikace, napsané v Nette Frameworku, sadu konzolových příkazů, které je pak možné spouštět přes příkazový řádek (terminál).

Řada použitých Nette knihoven z projektu Kdyby a dalších toto rozhraní využívá pro správu aplikace a její integrace je tak nezbytná. Navíc bude knihovna použita pro implementaci sady skriptů, které bude v daných intervalech vykonávat CRON. Typicky pak půjde např. o zasílání upomínkových emailů, periodické stahování plateb z banky atd. Díky této knihovně tedy bude možné psát obslužné skripty přímo v PHP v Nette Frameworku, a tedy bez potřeby dalších podpůrných skriptů napsaných v jiných jazycích (např. Bash, Python ad.).

7.1.2.4 Kdyby/Events

Nette Framwork sice nativně implementuje základní podporu pro generování a zachytávání událostí, nicméně posluchače událostí je potřeba registrovat přímo na konkrétní instanci entity, která události generuje, a nelze moc upravovat pořadí jednou registrovaných posluchačů, tj. určit pořadí, v jakém mají být obsluhování.

Systém pro práci s událostmi je ale stěžejní pro modulární návrh aplikace. Proto bude použita knihovna Kdyby/Events [21], která (mimo jiné) tento systém obaluje a rozšiřuje o možnost registrovat

[28] MVP – Model View Presenter.

posluchače událostí v místě obsluhy události, tedy ne v místě vzniku události, a rovněž umožňuje určit pořadí (prioritu), v jakém mají být jednotlivé obsluhy volány.

7.1.2.5 Kdyby/Doctrine

Nette framework sice obsahuje knihovnu pro komunikaci s MySQL a dalšími databázemi, nicméně tato knihovna je velmi jednoduchá, neumožňuje složitější druhy dotazů a pro větší projekty je tak nevhodná. Z tohoto důvodu bude použita knihovna Kdyby/Doctrine [22], která integruje ORM[29] knihovnu Doctrine [23] do Nette Frameworku a dále ji rozšiřuje o další přidanou funkcionalitu.

Doctrine patří mezi nejpopulárnější profesionální knihovny pro práci s databázemi pro PHP současnosti, disponuje skvělou dokumentací a obrovskou komunitou.

Mezi přednosti Doctrine patří hlavně její vlastní jazyk DQL[30], který se pak následně v závislosti na použité databázi překládá do jazyka, kterému databáze rozumí, velké množství databází, které Doctrine nativně podporuje, vysoká míra abstrakce a propracovaný ORM model. Na životní cyklus ORM entit lze rovněž navázat posluchače událostí a vykonávat tak různé akce např. při ukládání nebo mazání záznamů z databáze. Samozřejmostí je pak generování nebo úprava schématu databáze na základě ORM entit, nebo naopak vygenerování ORM entit na základě databázového modelu či XML nebo *YAML* konfigurace.

Dalšími důvody pro použití Doctrine jsou pak rozšíření *Doctrine-migrations* a *Doctrine-behaviours*, které do Nette Frameworku integruje projekt *Zenify* (viz dále).

7.1.2.6 Zenify/Doctrine-migrations

Zenify/Doctrine-migrations [24] rozšiřuje Doctrine o možnost generování migračních skriptů. Při změně ORM schématu je toto rozšíření schopné na základě aktuálního schématu databáze vygenerovat posloupnost příkazů, které je potřeba nad databází vykonat tak, aby se tato schémata sjednotila. V jednoduchosti to pak znamená, že pokud je přidán k ORM entitě uživatele další atribut, reprezentující např. datum narození, rozšíření automaticky vygeneruje pro databázi migrační skript, který do databázového schématu tento atribut přidá.

Generované migrační skripty obsahují jak migrace pro migraci nahoru (aktualizaci na novou verzi) tak dolů (vrácení k předchozí verzi), díky čemuž je možné se časem vrátit k původnímu schématu databáze před spuštěním migrace. Migrační skripty pak lze libovolně upravovat a přidávat i další sady příkazů, které je potřeba nad databází provést. Generované skripty totiž zajišťují pouze migraci schématu databáze, nikoliv dat. Tuto obsluhu si musí programátor dopsat sám.

Zenify/Doctrine-migrations je rovněž kompatibilní s Kdyby/Console, díky čemuž lze generování migrací a jejich spouštění pohodlně vykonávat z příkazového řádku.

7.1.2.7 Zenify/Doctrine-behaviours

Zenify/Doctrine-behaviours [25] je další rozšíření pro Doctrine, díky kterému je možné ORM entity snadno rozšířit o další funkcionalitu a které využívá hlavně životních cyklů ROM entit. Mezi nejzajímavější funkcionalitu, kterou je plánováno použít pak patří:

- možnost nepřímého mazání entit, kdy záznam není z databáze trvale smazán, ale je mu pouze nastaven příznak s datem smazání;
- časová značka, která při manipulaci s entitou (vytvoření / editace) zaznamenává datum a čas změny;
- zaznamenání uživatele, který entitu vytvořil, naposledy upravil a smazal;
- logování změn entity (záznamu), díky čemuž lze vést u požadovaných entit audit.

Dále pak knihovna implementuje možnost udržovat verze entit, entity lokalizovat (překládat) a další pokročilou funkcionalitu.

---

[29] ORM – Object Relation Mapping, programovací technika zajišťující konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem.

[30] DQL – Doctrine Query Language.

7.1.2.8 Wiredmedia/Doctrine-extension

Problémem s Doctrine, potažmo s jazykem DQL je, že byl navrhnut tak, aby byl obecný a jednotný pro všechny druhy relačních databází, v důsledku čehož nativně postrádá některé pokročilejší funkce. Proto bude do projektu začleněna i knihovna Wiredmedia/Doctrine-extension [26], která do jazyka DQL přidává základní funkce pro práci s datem, časem, textem a aritmetické funkce.

7.1.2.9 TIESA/Ldap

Open-Source PHP knihoven pro práci s LDAP databází je velké množství. Pro účely této práce jsem vybral knihovnu TIESA/Ldalp [27]. Jedná se o malou Open-Source knihovnu pro základní práci s LDAP databází, jež je velmi intuitivní a jednoduchá na použití.

7.1.2.10 TCPDF

TCPDF [28] je velmi všestranná Open-Source knihovna pro generování PDF souborů. PDF soubory je možné generovat buďto přímo pomocí volání jednotlivých funkcí pro vkládání a stylování obsahu, nebo je lze generovat na základě HTML a CSS kódu, což je podstatně jednodušší a přívětivější varianta (obzvláště v případě implementace webové aplikace). Při generování souborů z HTML je ovšem potřeba počítat, že knihovna nepodporuje všechny standardní CSS styly, čímž je tato funkce značně limitována, nicméně pro účely této práce (generování registračních formulářů) by měla bohatě stačit. Knihovna dále podporuje vkládání obrázků a dalšího multimediálního obsahu, generování 1D, 2D a QR kódů, úpravu záhlaví a zápatí dokumentu a mnoho dalšího.

Na stránkách vývojářské komunity je pak k dispozici detailní dokumentace spolu s více jak 60 příklady použití jednotlivých funkcí.

7.2 Uživatelské role a systém oprávnění

Nette Framework nativně obsahuje základní autentifikační a autorizační systém, který umožňuje definování uživatelských rolí, jež od sebe mohou dědit a pro něž pak lze povolit nebo zakázat jednotlivé prostředky a akce, které nad nimi mohou vykonávat [29].

Vzhledem k povaze výsledného systému bude přístup k funkcionalitě určován na základě činnosti (role), kterou bude člen (uživatel) pro klub vykonávat (viz uživatelské role definované v kapitole 4.1). Samotnou funkcionalitu pak budou zabezpečovat jednotlivé moduly, které by tak měly samy zabezpečit kontrolu přístupu k nabízené funkcionalitě, tj. definovat vlastní sadu uživatelských rolí a oprávnění.

Proto bude systém implementovat pouze grafické rozhraní pro přiřazení role uživateli, bez možnosti definovat vlastní uživatelské role či upravovat přístup k dostupné funkcionalitě.

Z těchto důvodů nebyl uvažován pokročilejší systém pro správu oprávnění a pro potřeby aplikace bude využit nativní bezpečnostní systém dostupný v Nette Frameworku.

7.2.1 Autentifikace uživatelů

Z důvodu zvýšení bezpečnosti celého systému budou hesla uživatelů uložena v databázi pouze v zašifrované formě pomocí jednocestné šifry s přidanou, náhodně generovanou solí. Z tohoto důvodu bude nejprve vyhledán uživatel na základě poskytnutého uživatelského jména (případně UID nebo systémového emailu) a v případě nalezení unikátního záznamu bude porovnán hash uloženého hesla s hashem hesla zadaného při přihlášení uživatele. V případě úspěšné autentifikace bude uživateli umožněn vstup do aplikace a vytvořena relace identifikující přihlášeného uživatele společně s výčtem jeho rolí. V případě neúspěšné autentifikace bude uživateli zobrazena vždy stejná chybová hláška bez ohledu, zdali zadal chybné uživatelské jméno nebo heslo (důvodem je znemožnění hádání uživatelských jmen).

7.2.2 Autorizace uživatelů

Vzhledem k malému množství uživatelských rolí a jejich pevně přidruženým úkonům se nepředpokládá omezování (či rozšiřování) dostupné funkcionality pro konkrétního uživatele. Stejně tak by měl každý modul (balíček funkcionalit) využívat buď vlastní definované role, nebo role jádra

systému. Z těchto důvodů se nepředpokládá potřeba řídit přístup k funkcionalitě na základě konkrétních zdrojů a akcí, a proto bude probíhat kontrola oprávnění pouze na základě uživatelských rolí.

Samotná kontrola oprávnění pak bude probíhat jednak v presenteru (ověření, že uživatel může přistupovat k dané stránce s funkcionalitou) a jednak u každé dostupné funkcionality na začátku obsluhy každé akce (tj. vytvoření, úprava, mazání), případně pak na dalších místech.

7.3 Modularita aplikace

Výsledná aplikace má mít modulární architekturu, díky čemuž se budou lépe spravovat a testovat její jednotlivé části. Jak bylo ale zjištěno při analýze existujících řešení (především u systému DUSPS, kapitola 6.2.1), tak členění aplikace do velkého množství modulů může být kontraproduktivní, jelikož se následně systém jako celek velmi špatně spravuje a rozšiřuje. Velká míra abstrakce a zejména špatný návrh pak může vývojáře nutit vytvářet mezi jednotlivými moduly pevné vazby, bez možnosti jejich následné separace, bez nutnosti zásahu do kódu aplikace.

Z tohoto důvodu jsem se rozhodl aplikaci rozdělit pouze na jádro a pár několik dalších modulů, u kterých se do budoucna předpokládá, že by mohly být ze systému vyňaty nebo nahrazeny moduly jinými. Zbytek hlavní funkcionality, která zároveň zabezpečuje základní provoz a funkci klubu, pak byla zahrnuta přímo do jádra aplikace.

7.3.1 Rozdělení do modulů

7.3.1.1 Modul Core

Bude obstarávat veškerou funkcionalitu pro správu členů (uživatelů), autentifikaci a autorizaci uživatelů, správu nastavení informačního systému, platební politiku členských příspěvků, nastavení aplikace a grafické rozhraní aplikace.

Do jádra bude rovněž začleněno řízení přístupu členů ke službám *počítačová sít* a *fitcentrum*. Důvodem integrace těchto služeb přímo do jádra a ne v podobě modulu je to, že provoz těchto služeb je pro klub existenčně zásadní z důvodu vybraných členských poplatků. Navíc hlavním smyslem informačního systému je řízení přístupu do počítačové sítě a separace této funkcionality tak z tohoto důvodu postrádá smysl. Pokročilá správa, především pak síťových prvků, ale může být do budoucna řešena pomocí vyjímatelných modulů.

7.3.1.2 Modul FioBank

Jádro bude sice obsahovat funkcionalitu pro platební politiku, přehled plateb, párování plateb k uživatelským účtům, nicméně komunikace s API konkrétní banky (aktuálně Fio) bude oddělena do samostatného modulu. Modul bude periodicky stahovat informace o proběhlých platbách a jádro systému pak notifikovat o nových platbách.

Důvodem pro oddělení je, že časem se může změnit API banky nebo může dojít k přechodu na jinou banku. Dalším důvodem je pak možnost integrace více platebních systémů najednou (například online platební brány).

7.3.1.3 Modul LDAP

LDAP modul bude zajišťovat komunikaci s LDAP databází systému a propagaci nastavení uživatelů z jádra aplikace do LDAP databáze. Důvodem k oddělení této funkcionality je možnost dočasně nebo trvale odstavit napojení LDAP databáze ať už z důvodu lokálního testování aplikace, nebo napojení infrastruktury počítačové sítě na jiné systémy.

7.3.1.4 Modul Mail

Modul pro emaily bude rozesílat veškeré emailové notifikace jádra aplikace. Důvodem k oddělení této funkcionality je možnost dočasného nebo trvalého odpojení této funkcionality. Dalším důvodem pro oddělení této funkcionality je pak demonstrace návrhu modulárního systému.

Mail modul bude integrovat novou položku do hlavního menu, komponentu pro nastavení SMTP serveru do stránky s nastavením informačního systému, vlastní URL adresy, ORM entity, konzolový interface a registrovat posluchače událostí a vlastní jazykové balíčky.

7.3.1.5 Modul API

Posledním modulem implementovaným v rámci této práce bude REST API pro zaznamenávání příchodů a odchodů z fitcentra společně s výpisem uživatelů, kteří mohou fitcentrum využívat a kteří jsou aktuálně ve fitcentru přítomni.

Důvodem pro implementaci je napojení vrátenské aplikace na výsledný informační systém. Navíc klub do budoucna zvažuje instalaci vlastního kartového systému na vstupní dveře fitcentra, v důsledku čehož může být tento modul zaměněn za modul spravující evidenci čipových karet a řízení přístupu do místností.

7.3.2 Integrace modulů

Integrace modulů do aplikace bude probíhat na základě jejich registrace v konfiguračním souboru aplikace, tedy obdobně jako je tomu u běžných Nette rozšíření.

Pro jednotlivé moduly pak bude v jádru aplikace implantováno rozhraní, pomocí kterého bude modul registrovat vlastní URL adresy (na vlastní presentery), odkazy na jazykové soubory, definice ORM entit a další funkcionalitu.

Integrace do grafického rozhraní bude rovněž řešena pomocí konfiguračních souborů. Hlavní (postranní) navigace bude definována v konfiguračním souboru jádra, kterou pak mohou další konfigurační soubory jednotlivých modulů rozšiřovat o další položky. Integrace do ostatních UI komponent bude řešena pomocí uvedení patřičného *tagu* u registrace továrny komponenty (např. pro integraci komponenty pro nastavení emailového modulu do stránky s nastavením aplikace bude potřeba u továrny této komponenty nastavit tag „*admin.core.setting.panel*“).

7.3.3 Komunikace mezi moduly

Jelikož bude aplikace modulární, je potřeba zajistit způsob komunikace mezi jednotlivými moduly (předávání zpráv). Jednou z možností je předávat informace přímo, nicméně takovýto model obvykle vede k implementaci oboustranné závislosti modulů, což je nežádoucí, jelikož moduly pak od sebe musí být separovány v kódu, tj. bez možnosti modul dočasně nebo trvale z aplikace vyjmout.

Další možností by mohla být například implementace nebo využití nějakého již existujícího systému pro zasílání zpráv, což je ale v tomto případě zbytečně složité řešení, jelikož většinou bude potřeba reagovat pouze na konkrétní události, a to navíc uvnitř jediné aplikace. Služby pro předávaní zpráv tak mají spíše význam pro zajištění komunikace mezi oddělenými systémy.

Pro potřeby této práce tak bude použit systém generovaných událostí, kdy jednotlivé služby a komponenty aplikace budou při vykonání akce (většinou při vytvoření, úpravě a smazání záznamu) generovat události s informací o provedené akci.

Nette navíc nativně registruje velké množství událostí, přičemž mezi ty nejzajímavější z hlediska modulárního návrhu nejspíše patří události generované při odeslání formuláře nebo změně stavu uživatele (přihlášení, odhlášení ad.). Další výhodou je pak možnost napojení na události životního cyklu Doctrine entit, případně dalších integrovaných knihoven.

7.4 Datový model

Návrhů, jak a do jaké míry (kolika vrstev) členit datový model existuje mnoho. Někteří vývojáři člení datový model minimálně (na vrstvu ORM a vrstvu obsluhy), případně vůbec, někteří pak používají i 5 a více vrstev, kdy pro jednotlivé ORM entity vytvářejí i tzv. datové kontejnery, se kterými pak komunikují navenek mimo datový model aplikace. Opa přístupy pak mají své pro i proti. Malé členění znamená rychlý vývoj, ale rovněž nepřehledný kód, ve kterém lze snadno udělat množství chyb. Velké členění pak vetší míru abstrakce, snazší záměnu některých součástí, lepší

testování a čitelnější kód nicméně za cenu náročnějšího návrhu a implementace, a implementaci velké spousty zbytečné funkcionality, která pouze přenáší data z jedné vrstvy do druhé.

Osobně se mi osvědčilo datový model členit do 3 vrstev, pokud pomineme vrstvy, které zajišťuje databázová vrstva (Doctrine knihovna), a to na vrstvy ORM, Fasády a Služby, přičemž toto členění by mělo být pro potřeby této práce dostačující.

7.4.1 ORM

ORM vrstva představuje jednotlivé entity datového modelu. Entity mají atributy, definují vazby mezi sebou a obsahují základní logiku v podobě sanace a kontroly vstupních dat tak, aby se entita nedostala do nevalidního stavu. Navenek pak entita disponuje rozhraním pro její modifikaci.

Někteří vývojáři do entit přenášejí i pokročilejší logiku, která zahrnuje vazby mezi entitami a další mechanizmy. Mne osobně se tento princip moc neosvědčil, jelikož v některých případech, např. při dotazu na databázi obsahující spojení tabulek a různá kritéria, může být z databáze získána pouze podmnožina všech atributů ORM betit a vazeb na ostatní entity. Pokud pak s tímto chováním datového modelu vývojář při návrhu nepočítá nebo jej opomene, může se entita ocitnout v nevalidním stavu. Navíc tento model pak může vývojáře svádět k tomu, aby do ORM entit zaintegrovali i další databázové vrstvy a v kódu se pak lze špatně vyznat.

7.4.2 Fasády

Fasády budou v aplikaci sloužit jako mezivrstva mezi službami a databázovou vrstvou, budou zajišťovat generování a vykonávání DQL dotazů a perzistenci ORM entit. V jistém ohledu budou obdobou repositářů ORM entit s tím rozdílem, že budou moci přistupovat i ke spojovým ORM entitám, které zajišťující rozšířenou relaci dvou či více entit.

7.4.3 Služby

Poslední vrstvou datového modelu budou služby, které budou zajišťovat komunikaci databázového modelu se zbytkem aplikace a zbytek logiky datového modelu. Každá dostupná funkcionalita služby pak bude pomocí *Fasád* vykonávat sady DQL dotazů a zajišťovat jejich transakční zpracování.

Komunikace mezi službami (a moduly) by pak měla probíhat hlavně pomocí generovaných událostí. V základu bude každá služba generovat alespoň události *beforeCreate, beforeUpdate, beforeDelete* a *afterCreate, afterUpdate, afterDelete*. *Before* události budou sloužit primárně pro možnost další validace entit nebo zahrnutí dalších změn do transakce v návaznosti na změnu entity (např. při povolení služby *počítačová síť* musí zároveň dojít k přiřazení VLANy, jinak nelze akci úspěšně provést). *After* události pak budou sloužit k oznámení úspěšně provedené akce po dokončení transakce a využít je bude možné například pro odeslání potvrzujícího emailu nebo propagaci do LDAP databáze.

7.4.4 Schéma databáze

Schéma databáze bylo navrhnuto s ohledem na formální normy, nicméně v některých případech byly tyto normy porušeny z důvodu zjednodušení SQL dotazů a zrychlení práce nad databází. Dalším důvodem pro porušení normy pak bylo zjednodušení práce s Doctrine knihovnou, kde se celkově špatně manipuluje s kompozitními identifikátory tabulek. Každá tabulka aplikace tak má jako jedinečný identifikátor sloupec *id*.

Schéma návrhu databáze je k dispozici v příloze (Obrázek 4). Vzhledem k přehlednosti byly z vizualizace odstraněny čáry představující relace *created_by, updated_by,* a *deleted_by* na tabulku *core_user*.

Dále budou v krátkosti představeny jednotlivé tabulky informačního systému.

7.4.4.1 Modul Core

7.4.4.1.1 Tabulka core_person

Obsahuje veškerá osobní data uživatelů držená v informačním systému. Důvodem k oddělení těchto dat od tabulky *core_user* je plánované napojení na centrální databázi členů SU ČVUT.

7.4.4.1.2 Tabulka core_user

Obsahuje autentifikační a autorizační informace o uživatelích společně s preferovanou lokalizací uživatele. Důvodem oddělení od tabulky *core_member* je potenciální možnost udělit do informačního systému přístup i nečlenům klubu (např. implementace rozhraní pro vrátné nebo jiné druhy zvláštních uživatelů).

7.4.4.1.3 Tabulka core_member

Obsahuje evidenci všech členů klubu spolu se stavem jejich členství, datem registrace a s přepočítanými hodnotami, jako je datum následující úhrady členského poplatku a stavy služeb klubu (povoleno, zakázáno, trest).

7.4.4.1.4 Tabulka core_log

Tabulka zajišťující audit některých tabulek (ORM entit). Nese informace o typu změny (vytvoření, úprava, smazání), identifikaci entity, provedené změně (původní a nový stav) a identifikaci uživatele, který změnu provedl společně s časovou značkou.

7.4.4.1.5 Tabulka core_transaction

Obsahuje veškeré registrované platby a jejich základní informace, jako je druh platby, částka, měna, variabilní symbol, přiřazení ke členovi a případně další přidružené informace o platbě.

7.4.4.1.6 Tabulka core_setting

Obsahuje globální nastavení systému, které je možné editovat v grafickém rozhraní aplikace.

7.4.4.1.7 Tabulka core_network_device

Slouží pro vedení evidence všech síťových zařízení členů klubu společně s MAC adresou a popisem zařízení.

7.4.4.1.8 Tabulka core_network_ban

Slouží pro vedení evidence síťových trestů uživatelů společně s odůvodněním trestu a délkou trvání.

7.4.4.1.9 Tabulka core_network_vlan

Slouží pro evidenci dostupných VLAN sítí spolu s možnými přidruženými informacemi.

7.4.4.1.10 Tabulka core_member_to_vlan

Spojová tabulka pro relaci mezi tabulkou *core_network_vlan* a *core_member*. Tabulka obsahuje i časovou značku odkdy do kdy byla uživateli VLAN síť přiřazená.

7.4.4.1.11 Tabulka core_gym_ban

Slouží pro vedení evidence trestů uživatelů fitcentra společně s odůvodněním trestu a délkou trvání.

7.4.4.1.12 Tabulka core_gym_access_log

Slouží pro vedení evidence data a času vstupů a odchodů členů z fitcentra.

7.4.4.2 Modul FioBank

Tento modul obsahuje pouze jedinou tabulku a to *fio_bank_transaction*, která uchovává všechny transakce získané z Fio banky v původní podobě a slouží k jednoznačné identifikaci plateb, které již byly registrovány a předány jádru aplikace. Tato tabulka navíc drží další dodatečné informace o platbách, které bylo možné z Fio banky získat.

7.4.4.3 Modul Mail

Tento modul obsahuje pouze jedinou tabulku, a to *mail_queue*, která slouží jako fronta pro odchozí emaily hromadně zasílané informačním systémem (více v kapitole 7.7).

7.4.4.4 Další tabulky

Další tabulky mohou registrovat různá rozšíření Nette Frameworku, případně jiné PHP knihovny. Systém bude s největší pravděpodobností obsahovat pouze jednu takovouto tabulku, a to tabulku *doctrine_migrations*, která bude obsahovat evidenci provedených migrací.

7.5 Napojení na Fio banku

Napojení na Fio banku bude realizováno pomocí modulu z důvodu případné budoucí změny banky, nebo integrace dalších platebních systémů do aplikace.

Pro účely aplikace bude potřeba v předem definovaných intervalech stahovat informace o nových transakcích. K tomuto účelu vystavuje Fio banka jednoduché API v podobě GET dotazu na patřičnou URL adresu. URL adresa vyžaduje uvedení přístupového tokenu, období, za které chceme získat transakce (datum od, do), a požadovaný formát odpovědi (JSON, CSV, XML, HTML a mnoho dalších).

Systém bude stahovat transakce ve formátu CSV, který lze v PHP velmi dobře zpracovávat. Platby budou stahovány v hodinovém intervalu za posledních 30 dní (z důvodu případného výpadku nebo odstávky systému, internetového připojení apod.). Stažené transakce budou uloženy v *temp/* adresáři a po úspěšném zpracování systémem (vložení do databáze) bude soubor s transakcemi smazán. V případě jakékoliv chyby při zpracování zůstane soubor zachován (z důvodu případné detekce chyby) a administrátorovi systému bude zaslán email oznamující chybu.

Pro účely periodického stahování transakcí bude systém implementovat konzolové rozhraní, které umožní CRONu nebo administrátorovi vynutit stažení nových transakcí.

Pro účely registrace již dříve zpracovaných transakcí a uchování originální historie transakcí budou všechny stažené transakce uchovány v tabulce *fio_bank_transaction*. Při registraci nové transakce bude rovněž notifikováno jádro informačního systému a transakce registrována v jádře systému, které se po splnění předem nastavených kritérií (výše platby, měna, odchylka ad.) pokusí podle variabilního symbolu přiřadit k transakci člena klubu. Platby v jádře (tabulce *core_transaction*) tedy budou obsahovat již zpracované a modifikované platby systémem nebo administrátorem systému, což je dalším důvodem dvojího ukládání transakcí (původní transakce získaná přes API banky a systémová transakce).

7.6 Propojení s LDAP

Napojení na OpenLDAP databázi bude implementováno pomocí zásuvného modulu, díky čemuž bude možné v případě potřeby (např. testování systému na lokálním PC) tuto funkcionalitu jednoduše deaktivovat odebráním modulu z konfigurace systému.

7.6.1 Synchronizace s LDAP databází

Modul bude poslouchat na řadě událostí jádra systému, konkrétně pak na událostech generovaných po vytvoření, aktualizaci nebo smazání osoby, uživatele, člena, členova síťového zařízení a změně přiřazené VLAN sítě. Při odchytu některé z uvedených událostí nejdříve LDAP modul získá patřičnou entitu z LDAP databáze, případně ji vytvoří, na entitu následně aplikuje veškeré nové nastavení a uloží ji.

Při výpadku nebo odstávce LDAP databáze bude potřeba synchronizovat LDAP a MySQL databázi. K tomuto účelu bude implementováno konzolové rozhraní pro možnost manuálního vynucení kompletní synchronizace těchto databází (resp. propagaci aktuálního stavu MySQL databáze do LDAP databáze). Tato synchronizace bude rovněž prováděna jednou denně pomocí naplánované CRON úlohy pro zajištění propagace konfigurace v případě výskytu neočekávaných chyb systému.

7.6.2 Schéma LDAP databáze

S návrhem DLAP databáze bohužel nemám téměř žádné zkušenosti. Jelikož LDAP databáze je stromová (souborová), chtěl jsem původně pro každou entitu (uživatel, zařízení, skupiny ad.) vytvořit vlastní uzel (entitu) a ty pak hierarchicky (stromově) uspořádat, díky čemuž by pak bylo možné do budoucna držet v LDAP databázi i další dodatečné informace (např. u zařízení popis, staticky přiřazenou IP atd.). Toto řešení se ale ve výsledku ukázalo jako nepraktické, jelikož LDAP databáze disponuje velmi omezeným dotazovacím jazykem, kdy není možné prohledávat entity pomocí podmínek vztahujících se k předku a potomku entity, nelze skládat pokročilejší druhy dotazů obsahujících sub dotazy apod. Technicky by sice bylo možné potřebné informace z databáze získat např. pomocí série dotazů, kdy by nejdříve byl vyhledán uživatel a následně dalším dotazem vyhledáno jeho konkrétní zařízení (pro ověření, že se daný uživatel může s daným zařízením připojit do sítě), nicméně většina systémů, které umožňují napojení na LDAP databázi, s tímto druhem dotazů a ověřování nepočítá.

Proto jsem se rozhodl pro každého uživatele (člena) vytvořit pouze jednu entitu v LDAP databázi a v ní pak držet veškeré přidružené informace pomocí různých atributů. Níže budou vyjmenovány atributy a jejich účel:

- *accountStatus* – stav účtu (aktivní / neaktivní);
- *cn* – id LDAP entity v podobě uživatelského jména uživatele;
- *displayName* – celé jméno uživatele;
- *mail* – klubový email uživatele pod doménou hk.cvut.cz;
- *employeeNumber* – id osoby z MySQL tabulky *core_person* (pro potřeby synchronizace);
- *enabledServices* – obsahuje seznam povolených služeb (mail, smtp, network, gym ad.) a seznam přiřazených systémových rolí;
- *givenName* – jméno uživatele;
- *sn* – příjmení uživatele;
- *homePostalAddress* – adresa trvalého bydliště;
- *homeDirectory* – cesta k domovské složce (užívané pro PAM autentifikaci administrátorů na serverech);
- *macAddress* – seznam MAC adres zařízení registrovaných u uživatele;
- *mailForwardingAddress* – skutečná emailová adresa uživatele (mimo doménu hk.cvut.cz);
- *mailMessageStore* – cesta ke složce pro ukládání emailů (pokud má uživatel povolenou službu mailbox);
- *mailQuota* – velikost mailboxu (v bytech);
- *memberOfGroup* – seznam hromadných emailových aliasů, které jsou uživateli přiřazeny na základě přiřazených rolí (např. rada@hk.cvut.cz);
- *userPassword* – heslo uživatele, šifrované pomocí několika druhů šifer (kvůli různým službám);
- *preferredLanguage* – nastavený preferovaný systémový jazyk;
- *radiusTunnelPrivateGroupId* – číslo VLAN sítě, která je uživateli přiřazena;
- *sshPublicKey* – seznam uživatelových veřejných SSH klíčů (užívané pro PAM autentifikaci administrátorů na serverech);
- *telephoneNumber* – telefonní číslo uživatele (je-li evidováno);
- *uidNumber* – jedinečný identifikátor člena v rámci klubu (UID číslo);
- *uid* – uživatelská jména, pomocí kterých se může uživatel autentifikovat (UID, uživatelské jméno, email adresa v rámci klubu).

Jak si lze povšimnout, tak některé informace jsou duplikovány (např. uživatelské jméno, UID, email), což je dáno především napojenými systémy, které pro svou funkci a dotazy nad LDAP databází využívají různé atributy. Rovněž si lze povšimnout, že při návrhu bylo zohledněno i budoucí napojení dalších systémů, jako je například PAM autentifikace, klubová emailová schránka ad. Tyto systémy jsem sice již z vetší části zprovoznil, nicméně pro účely této práce nebudou dále uvažovány a do výsledného informačního systému nebudou ani integrována uživatelská rozhraní pro jejich konfiguraci.

7.7 Mail systém

Nette Framework obsahuje v základu mail systém s možností rozesílat emaily pomocí lokálního nebo vzdáleného mail serveru (přes SMTP). Zároveň nabízí velmi propracovanou komponentu pro jednoduché vytváření bohatých HTML emailů s podporou příloh, UTF-8 ad. kódování znaků a mnoho dalšího, což velmi usnadňuje generování a rozesílání emailových zpráv, jelikož nativní dostupné funkce PHP v tomto ohledu velmi zaostávají.

7.7.1 Mail fronta

Byť se rozesílání emailů jeví jako triviální záležitost, z praxe mám zkušenosti, že tomu tak není, obzvláště pak u systémů, u kterých je předpoklad, že budou rozesílat emailové zprávy ve větším množství. Emaily se většinou rozesílají jen zřídka, proto se zpravidla neoplácí trvale udržovat připojení k SMPT serveru v nějaké cache paměti aplikace, v důsledku čehož pak při každé uživatelské akci, která generuje rozesílání emailů, je potřeba připojení k SMTP serveru navázat znovu, což stojí určitý čas. V případě webových systémů má pak toto chování negativní dopad na dobu odezvy aplikace. Dalším problémem pak mohou být různé ochrany emailových serverů proti SPAMu, např. Google, který neumožňuje rozesílání mailů více jak několika desítkám uživatelů. Pokud je tak potřeba rozeslat více než 50, či stovky emailů, je nutné rozesílat emaily v menších dávkách (několik desítek kusů), mezi kterými pak je nutné dělat několikaminutové rozestupy.

Z těchto a dalších důvodů se mi proto osvědčilo implementovat do takových systémů emailovou frontu, do které jsou zařazeny veškeré emaily, které mají být rozeslány, a ty pak rozesílat v menších dávkách s několikaminutovými rozestupy. V případě úspěšného odeslání emailu je pak email z fronty odebrán, jinak je do fronty vracen a přeplánováno jeho odeslání o cca 20 až 60 minut. Pokud se email nepodaří odeslat ani po N-tém pokusu, je email z fronty odstraněn, zalogována příčina a notifikován administrátor, příp. vývojář systému.

Jelikož bude informační systém hlavně na začátku pololetí rozesílat personalizované připomínky o uhrazení členského poplatku, bude do informačního systému implementován systém mailové fronty, pro zrychlení a zvýšení spolehlivosti rozesílaných emailů.

7.7.2 Nastavení SMTP serveru

Do stránky s nastavením informačního systému bude integrována komponenta umožňující konfiguraci připojení k SMTP serveru a nastavení chování mailové fronty (velikost rozesílaných dávek, počet pokusů o odeslání a délka prodlevy mezi pokusy).

7.7.3 Grafické rozhraní modulu

Pro členy kolejní rady a administrátory informačního systému bude implementováno rozhraní pro jednoduché rozesílání klubových emailů. Toto rozhraní umožní specifikovat příjemce, adresu pro odpověď, předmět zprávy a text zprávy.

Pro vývojáře bude k dispozici URL adresa (*/mail/<název_šablony>/),* pomocí které si bude moci vývojář otestovat zobrazení všech rozesílaných emailů (emailových šablon), pro snazší ladění a stylování emailových šablon.

7.7.4 Konzolové rozhraní

Mail modul bude implementovat konzolové rozhraní s těmito funkcemi:

- odeslání testovací zprávy, pro účely testu nastavení SMTP serveru;
- odeslání připomínek o úhradě členských poplatků, pro vytvoření CRON úlohy;
- odeslání dávky emailů z fronty;
- zjištění stavu emailové fronty.

7.7.5 Přehled rozesílaných emailů

Informační systém bude rozesílat tyto emailové notifikace:

- notifikaci o změně (přidání, odebrání) systémových rolí;

- žádost o potvrzení emailu (při změně emailu uživatele);
- potvrzení registrace člena, spolu s informacemi pro první úhradu členského příspěvku;
- potvrzení přijetí úhrady členského poplatku;
- varování o přijetí příchozí platby, která nevyhověla zadaným parametrům, nebo se ji nepodařilo navázat na účet existujícího člena (správci systému);
- informace o změně přihlašovacích údajů (provedené správcem systému);
- připomínku úhrady členského poplatku.

7.8 Automatizace procesů

V zadání práce je požadavek o specifické zaměření na automatizaci procesů. Veškerá tato automatizace ale byla zahrnuta už do funkčních požadavků (viz kapitola 4), kde jsou i detailně popsány jednotlivé kroky některých procesů, a proto zde již nebudou znovu uvedeny.

Drtivá většina automatizovaných úloh bude prováděna ihned v návaznosti na uživatelskou nebo systémovou akci. Zbylé úlohy, které je potřeba provádět periodicky v předem definovaných intervalech, budou implementovány pomocí konzolového interface informačního systému a budou automaticky spouštěny jako CRON úlohy. Konkrétně pak půjde o tyto úlohy:

- odesílání mailů z mail fronty, každé 2 minuty;
- deaktivace účtu při neuhrazení členského poplatku, každou hodinu;
- reaktivace síťové služby po vypršení trestu, každou hodinu;
- reaktivace služby posilovna po vypršení trestu, každou hodinu;
- stažení nových bankovních transakcí, každou hodinu;
- vyvážení přiřazených VLAN sítí (aby byla každá VLAN síť přiřazená ideálně jen jednomu uživateli), každou hodinu;
- rozeslání upomínky o úhradě členského poplatku, jednou za den;
- generování seznamu uživatelů s povoleným MAC-bypass, každých 15 minut;
- registrace odchodu člena z posilovny (pokud je v posilovně déle jak 4 hodiny nebo po půlnoci, tj. vrátenská služba nezaevidovala odchod člena z posilovny), každých 15 minut;
- deaktivace všech uživatelů, kteří nedoplnili do nového systému dodatečné osobní informace, každých 15 minut;
- synchronizace LDAP databáze, jednou za den;
- záloha MySQL databáze, jednou za den.

7.9 Generování reportů

V současnosti je potřeba generovat pouze jediný report, a to report přikládaný ke zprávě o stavu klubu, který obsahuje seznam členů za daný kalendářní půlrok spolu s výší vybraného členského poplatku.

Pro tento report jsem se rozhodl zvolit formát CSV, který lze generovat velmi jednoduše pomocí nativních funkcí PHP, je podporován aplikacemi Microsot Office, Libre Office, a Open Office, tj. většinou běžně používaných kancelářských aplikací. Zároveň je tento formát lidsky čitelný, tj. lze jej číst i bez použití speciální aplikace v obyčejném textovém editoru.

Uživatelské rozhraní aplikace bude umožňovat generování tohoto reportu podle zadaného časového úseku a reporty tak bude možné generovat i zpětně.

7.10 Tisk dokumentů

Klub musí archivovat registrační formuláře svých členů v papírové podobě, zároveň je ale potřeba zanášet stejné osobní informace z registračního formuláře do informačního systému, což je nepraktické a práce je tak zdvojená. Z tohoto důvodu jsem do funkčních požadavků na nový systém vložil i požadavek na tisk registračních formulářů a složenek na úhradu členských poplatků. Díky tomu se pak celý proces registrace urychlí a nový člen bude muset zadávat osobní informace pouze

jednou, do registračního formuláře v informačním systému, jenž pak následně vygeneruje a vytiskne registrační formulář se zadanými daty, který pak nový člen pouze podepíše.

Abych tomuto požadavku vyhověl, snažil jsem se nejdříve nalézt nějakou univerzální PHP knihovnu, která by umožňovala např. pomocí IPP[31], LPR[32], nebo jiného protokolu tisknout dokumenty přímo na tiskárně. Nakonec se tato snaha ukázala jako zcestná, jelikož veškeré knihovny umožňovaly tisk pouze v základním textovém režimu bez pokročilejších možností text stylovat a upravovat. Tyto knihovny lze použít např. pro tisk různých paragonů a dalších obdobných dokladů, ale pro tisk dokumentů jsou nevhodné. Nakonec jsem pak usoudil, že nejjednodušším řešením bude nainstalovat na webový server CUPS[33], který umožňuje generovat a plánovat tiskové úlohy na základě zaslaných PDF a dalších dokumentů.

Dalším krokem tak bylo generování PDF dokumentů (registračních formulářů a složenek) s osobními informacemi. Pro tento účel jsem chtěl vytvořit šablony PDF souborů s formulářovými prvky, do kterých by aplikace pouze doplnila potřebné údaje a výsledný dokument pak odeslala tiskovému serveru ke zpracování. Při experimentech jsem ale nebyl schopen pomocí žádné PHP knihovny nebo Linux aplikace vkládat do formulářových prvků text s diakritikou. Při bližším zkoumání a rešerši na internetu jsem pak zjistil, že tento problém je znám a s největší pravděpodobností je zaviněn chybnou implementací knihovny *pdftk*.

Vzhledem k tomu, že se mi tento problém nepodařilo ani po několikadenním snažení uspokojivě vyřešit, rozhodl jsem se generovat celé PDF soubory za použití knihovny TCPDF pomocí HTML a CSS stylů a Open-Source UTF-8 fontů. Pomocí PHP funkce *exec* pak bude následně spuštěna aplikace *lpr*, která z PDF vygeneruje tiskovou úlohu a odešle ji na tiskový server CUPS.

7.11 Zajištění redundance

Způsobů pro zajištění redundance webové aplikace je více, jedním z nich je pak implementace horizontálně škálovatelného systému s předřazeným loadbalancerem, který zároveň zajistí rozložení zátěže mezi více strojů, a tedy i zvýšení výkonu systému. Implementace takových to systémů ale vyžaduje pokročilejší znalosti návrhu a vývoje aplikací, a zároveň větší technickou zdatnost pro síťové administrátory.

Vzhledem k tomu, že se neočekává vysoké zatížení aplikace, a není tak potřeba řešit horizontální škálovatelnost, a také kvůli snaze udržet výslednou aplikaci pokud možno co nejjednodušší a snadno rozšiřitelnou i bez znalosti pokročilých programovacích technik, rozhodl jsem se zajistit redundanci aplikace na vrstvě operačního systému.

Dle nefunkčních požadavků budou k dispozici 2 instance virtuálních serverů na dvou oddělených strojích. Na tyto instance bude nainstalován GlusterFS[34], pomocí kterého bude vytvořen sdílený diskový prostor, na kterém se budou nacházet zdrojové soubory informačního systému a soubory MySQL a OpenLDAP databáze. Dále bude na obou instancích nainstalován keepalived[35], který v případě výpadku hlavní instance přiřadí veřejnou IP adresu záložní instanci.

Díky tomuto návrhu by pak měla být zajištěna redundance informačního systému a jeho databází napříč oběma instancemi. Pro zajištění přenosu SESSION (uživatelské relace) mezi instancemi budou SESSION aplikace ukládány do Memcached[36], která je součástí standardní distribuce PHP a umožňuje replikaci mezi více strojů.

---

[31] IPP – Internet Printing Protocol.
[32] LPR – Line Printer Protocol.
[33] CUPS – Common UNIX Printing Systém, modulární unixový print server.
[34] GlusterFS – distribuovaný Open-Source souborový systém.
[35] Keepalived – směrovací software zajišťující rozložení zátěže a redundanci služeb.
[36] Memcached – distribuovaný systém mezipaměti, který drží informace v RAM paměti počítače.

7.12 Zhodnocení

V této kapitole byly v krátkosti přestaveny technologie, knihovny a frameworky, které budou použity při implementaci výsledného systému spolu s odůvodněním, proč byly vybrány a k jakému účelu budou sloužit. Dále byl představen návrh řešení systému uživatelských rolí, emailový systém a návrh datového modelu spolu s krátkým popisem schématu MySQL a LDAP databáze. Na závěr pak bylo představeno řešení některých největších technických problémů, které bylo potřeba při návrhu aplikace vyřešit, mezi které pak patří modularita systému a tisk dokumentů.

U většiny navrhovaných systémů byly rovněž debatovány i možná alternativní řešení společně s odůvodněním, proč nebyly vybrány pro návrh výsledného systému.

I když je tato kapitola obsáhlejší, doufám, že pomůže k lepšímu pochopení základních principů, na kterých je výsledná aplikace navržena, a případně poslouží jako podklad pro další vývoj tohoto systému. V následující kapitole proto budou představeny některé implementační detaily, pro lepší pochopení organizace struktury aplikace a způsobu, jakým byla sestavena.

8. Implementace

V této kapitole se blíže podíváme na strukturu projektu, způsob, jakým byly navrženy a implementovány komponenty grafického rozhraní, a nástroje, které byly použity pro sestavení výsledné aplikace. Na závěr pak bude v krátkosti představen způsob, jakým byla aplikace nasazena do produkčního prostředí.

8.1 Struktura projektu

Rozvržení implementační struktury projektu a jeho modulů kopíruje adresářová struktura projektu, znázorňuje Obrázek 5 a Obrázek 6 v příloze této práce. Podle adresářové struktury jsou pak tvořeny i jmenné prostory jednotlivých implementovaných objektů (částí kódů) aplikace. Dále proto bude v krátkosti tato adresářová struktura představena.

8.1.1 Základní struktura projektu

Assets obsahuje veškeré fonty, obrázky, implementované LESS styly a JS kód, spolu s dalšími JS knihovnami, které se nenacházejí ve standardních Bower[37] repositářích. Adresáře *bower_components, nbproject, node_modules* jsou generované vývojovými nástroji, a proto nebudou v příloze této práce. Zbylé soubory v kořenovém adresáři jsou pak konfigurační soubory pro vývojové nástroje.

Adresář *config* obsahuje veškeré konfigurační soubory aplikace, které byly kvůli přehlednosti rozděleny do několika dalších souborů. V souboru *extensions* lze nalézt konfigurační soubory všech integrovaných Nette rozšíření, přičemž pro každé rozšíření je vytvořen vlastní konfigurační soubor. Konfigurační soubory jádra a modulů aplikace se pak nachází v adresáři *modules*. Všechny tyto a další konfigurační soubory jsou pak importovány do souboru *config.neon*, ve kterém pak lze zakomentováním příslušného řádku s importem konfigurace modulu tento modul zakázat. Soubor *config.neon* pak dále obsahuje konfiguraci PHP, HTTP hlaviček, délku trvání relace (SESSION), mapování jmenného prostoru presenterů a další konfiguraci aplikace. Pro potřeby lokální úpravy konfigurace (pro konkrétní stroj) pak slouží soubor *config.local.neon*.

Adresář *modules* (v adresáři *src/app*) obsahuje jádro a další moduly aplikace.

Adresář *fonts* obsahuje fonty pro generované PDF dokumenty. Do adresáře *log* se zapisují veškeré logy a chyby, ke kterým došlo při běhu aplikace. V *migrations* pak lze nalézt všechny migrační skripty MySQL databáze.

Adresář *rsync* obsahuje konfiguraci pro DNS, DHCP a RADIUS server, která je v 15-ti minutových intervalech synchronizována s konfigurací na těchto serverech. Byť samotný informační systém neumožnuje tuto konfiguraci upravovat přes grafické rozhraní, přišlo mi příhodné zde tuto konfiguraci přesunout a mít tak jednotné místo pro úpravu těchto konfigurací napříč všemi virtuálními stroji. Samotné konfigurační soubory nebudou k práci přiloženy.

Adresář *temp* obsahuje veškeré dočasné soubory generované Nette, knihovnami a systémem. Ve *vendor* se pak nachází veškeré PHP knihovny, Nette Framework i jeho rozšíření. Ve složce *www* je pak veškerý obsah, který je veřejně přístupný, tj. zkompilované a minimalizované LESS styly (do formátu CSS), JS kód a knihovny, obrázky, fonty a *index.php*, na který pak webový server směřuje veškerou ostatní komunikaci.

8.1.2 Struktura modulu

Pro demonstraci bude představena pouze adresářová struktura jádra systému, které je ze všech modulů nejobsáhlejší.

[37] Bower – balíčkový manažer pro vývoj webových aplikací.

Adresář *lang* obsahuje veškeré textové překlady (v českém a anglickém jazyce). *DI* obsahuje kód pro registraci modulu do systému, v *Entity* jsou pak definovány kontejnery pro zjednodušení manipulace a validace některých druhů dat (Adresa, telefonní číslo ad.).

Adresář *ORM* obsahuje datový model a logiku aplikace, ve kterém jsou definovány ORM entity, Fasády, předdefinované některé často užívané vlastnosti ORM entit v podobě PHP trait (adresář *Property*), Služby a výjimky, které tyto služby mohou při zpracování požadavků generovat.

V *PDF* se nachází komponenty generující registrační formuláře a složenky, tj. dokumenty určené hlavně k tisku. Dále pak následují hlavně rozšířené části Nette o další funkcionalitu (routy, zabezpečení, UI a lokalizační systém). Třídy implementující konzolové rozhraní systému se pak nachází v adresáři *SymfonyConsole*.

View obsahuje veškerý kód týkající se grafického rozhraní aplikace, který pak byl rozdělen na veřejnou a neveřejnou část, přičemž neveřejná část se nachází v adresáři *Admin*. Tyto části jsou pak rozděleny na UI komponenty, presentery a HTML šablony presenterů.

8.2 Implementace vlastního modulu

Implementace vlastního modulu je velmi obdobná jako v případě implementace jakéhokoliv rozšíření do Nette Frameworku. Pro usnadnění práce byla implementována abstraktní třída *App\CoreModule\DI\ModuleExtension*, kterou stačí pouze rozšířit o integrovanou funkcionalitu modulu. Pro registraci ORM entit pak stačí implementovat metodu *getEntityMappings*, pro definici vlastních routovacích pravidel pak metodu *getRoutesDefinition* atd.

Každý modul by pak měl mít vlastní konfigurační soubor, ve kterém bude registrována implementace abstraktní třídy *ModuleExtension* a tento konfigurační soubor pak importován do konfiguračního souboru *config.neon*.

Na vnitřním uspořádání modulu pak moc nezáleží, nicméně i tak musí být dodrženo několik málo podmínek, v opačném případě bude hrozit potřeba upravit konfiguraci patřičných částí systému. Konkrétně pak jde o tato pravidla:

1) Jmenný prostor všech tříd by měl začínat prefixem *App\<název_modulu>*, v opačném případě pak hrozí konflikt s jinou implementovanou třídou aplikace nebo její knihovny.
2) Aby bylo Nette schopné určit potřebný presenter, musí jmenný prostor všech presenterů být ve formátu *App\<název_modulu>(\Admin)\Presenter\<název_prezenteru>Presenter* a dále musí modul definovat routy (viz např. implementace Mail modulu).
3) Soubory s překlady musí být umístěny v *locale* adresáři v kořenovém adresáři modulu.
4) Pokud bude modul implementovat komponentu pomocí předpřipravené třídy *App\CoreModule\UI\Control*, pak musí být v adresáři obsahující tuto implementaci HTML šablona s názvem *template.latte* pro tuto komponentu.

Zbytek chování aplikace (a tedy postupu implementace) by pak měl být totožný s běžným použitím Nette Frameworku.

8.3 UI komponenty

Většina projektů, se kterými jsem se v praxi setkal, se zpravidla vyznačovala dvěma přístupy k implementaci UI komponent a k nim přidružené logiky.

S první variantou se lze setkat hlavně u „líných“ programátorů (případně, pokud projekt tlačí čas), kdy je veškerá implementace UI prvků a jejich obsluhy přesunuta přímo do presenteru (příp. kontroleru) návrhového vzoru, což pak z pravidla vede k tomu, že je do presenteru injektováno obrovské množství v danou chvíli nepotřebných služeb, což může negativně ovlivnit výkon celé aplikace, a zároveň takovéto návrhy obsahují presentery o tisících řádků podpůrného kódu, ve kterém se nelze prakticky vyznat.

Uvědomělejší vývojáři si pak tento kód separují do různých komponent, nicméně i tak lze u těchto návrhů nalézt další nedostatky, kdy se pak do nich snaží nacpat pokud možno co nejvíce funkcionality a aplikační logiky. Ve výsledku pak např. jednoduchý registrační formulář integruje krom definice vstupních prvků a validace přijatých dat i obsluhu pro vytvoření uživatele, obsluhu pro odeslání emailové notifikace o registraci, obsluhu pro napojení na LDAP databázi, příp. další funkcionalitu, díky čemuž se výsledný kód stává opět hůře čitelný a tedy i spravovatelný, nemluvě o značném množství služeb, které je pak potřeba těmto komponentám předávat.

Jelikož mi oba tyto přístupy už jen z praktického hlediska připadají nevhodné, snažil jsem se celé grafické rozhraní (a vlastně i zbytek systému) členit na menší dílčí části, díky čemuž by měl být výsledný kód dobře čitelný a přehledný. Samotná komponenta pak obsahuje minimum kódu potřebného ke své funkci, např. v případě obsluhy tlačítka pouze zkontroluje, zdali je uživatel oprávněn tuto akci vykonat a v případě, že ano, provede sanaci vstupů a další řízení předá příslušné službě. Stejný postup byl aplikován např. při implementaci formulářů a dalších prvků, kdy formulář pouze vytvoří nebo upraví příslušnou entitu a tu pak předá službě k dalšímu zpracování, přičemž dále neřeší, jakým způsobem je pak s entitou manipulováno, nebo zdali na základě této akce došlo např. k odeslání emailové notifikace. Tento přístup navíc umožňuje větší znovu-využitelnost těchto komponent, kdy formulář pro vytvoření záznamu může být jednoduše použit i pro jeho editaci.

8.3.1 Předpřipravené komponenty

Pro usnadnění implementace další funkcionality jsem rozšířil některé standardní komponenty Nette, které se nacházejí ve jmenném prostoru *App\CoreModule\UI*.

Komponenta *Control* byla rozšířena o auto-injekci lokalizačního systému, který je tak nativně dostupný v implementaci komponenty a který je rovněž automaticky injektován do HTML šablony. Navíc komponenta automaticky předpokládá existenci HTML šablony *template.latte* v místě její implementace, díky čemuž odpadá opakovaná rutina tuto funkcionalitu do každé komponenty implementovat.

Dále bylo upraveno chování formulářů ve třídě *Form*. Tato třída nově implementuje metodu *createFormElements*, pomocí které by měly být definovány formulářové prvky společně s jejich validačními pravidly. Dále je k dispozici metoda *getDefaults*, jejíž implementace by měla vracet pole defaultních hodnot formuláře. Přepracován byl rovněž způsob zpracování formuláře, který nově implementuje metody *processSubmit* (voláno při odeslání formuláře), *processSuccess* (voláno při odeslání validního formuláře), *processError* (voláno při odeslání nevalidního formuláře nebo registraci chyby během fáze *submit* a *success*) a *processFinish* (voláno po dokončení zpracování veškerých procesů). Nativní implementace metody *processFinish* navíc při odeslání formuláře vynutí refresh stránky, čímž zabrání nechtěnému opětovnému odeslání duplicitního požadavku. V případě AJAX zpracování pak vyresetuje formulář. Samozřejmostí je pak auto-injekce lokalizačního systému obdobně, jako je tomu u ostatních komponent.

Co se dále týče formulářů, tak byly implementovány i speciální formulářové prvky pro zadávání data a času, emailu, MAC adresy a telefonního čísla, které nativně obsahují validační pravidla a v případě data, času a telefonního čísla přijímají a vracejí datové objekty určené pro jejich manipulaci.

Rovněž byl upraven *Presenter*, do kterého byla integrována podpora lokalizačního systému, implementována ochrana proti CORS útoku a veškeré odpovědi na AJAX dotazy převedeny na zasílání JSON odpovědí.

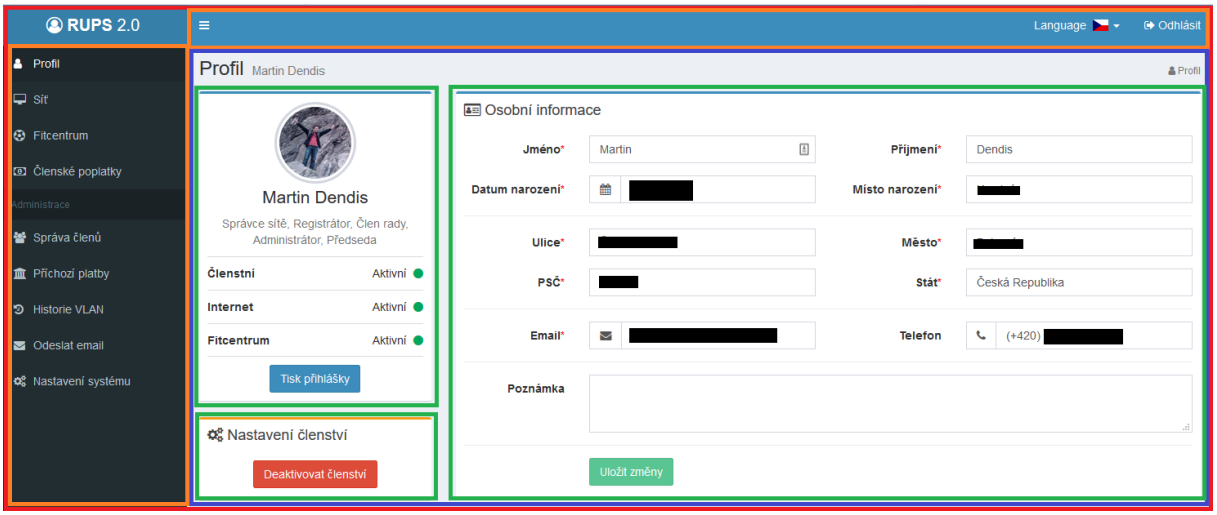
V poslední řadě pak byla implementována nová komponenta *OrderedLayout,* a k ní přidružený interface *IOrderableComponent*, které najdou uplatnění primárně u implementace rozšiřitelných (modulárních) komponent.

8.3.2 Členění UI

Pro lepší představu o členění grafického rozhraní přikládám snímek z aplikace (viz Obrázek 3), kde jsem barevně toto členění vyznačil.

Červeně je vytáhnut layout aplikace, který obsahuje horní a postranní menu, obsah stránky a patičku (v náhledu nejde vidět). V HTML šabloně layoutu jsou rovněž definovány všechny použité CSS styly a JS kódy aplikace. Horní a postranní menu pak tvoří jednotlivé komponenty, které lze přes konfigurační soubory rozšiřovat o další prvky (v případě horního menu pak komponenty).

Obsah (orámován modře) pak definuje šablona presenteru, do kterého jsou vloženy další komponenty (orámovány zeleně). Komponenta pro správu osobních informací je pak ještě implementačně rozdělena na část grafickou a formulář.



Obrázek 3 - Demonstrace členění UI

8.4 Sestavení aplikace

Pro sestavení výsledné aplikace byla použita řada vývojářských nástrojů, především je pak potřeba mít na stroji, na kterém bude aplikace sestavena, nainstalovaný NodeJS[38] a Composer[39]. V kořenovém adresáři projektu se nachází soubor *package.json*, který registruje potřebné utiliti pro sestavení JS, CSS ad. zdrojů, především pak Bower a Gulp[40]. Dále je k dispozici soubor *bower.json*, který registruje potřebné LESS, JS knihovny a fonty, soubor *gulpfile.js*, ve kterém jsou implementovány úlohy pro kompilaci a minimalizaci LESS a JS knihoven. Posledními důležitými soubory jsou pak *composer.json*, který registruje potřebné PHP knihovny, a soubor *composer.lock*, který obsahuje konkrétní verze PHP knihoven, které byly použité při sestavení projektu.

Při sestavení projektu je pak potřeba postupovat následovně:

1) Pomocí příkazu *npm install* provést instalaci Gulp a Bower utilit.
2) Pomocí příkazu *bower install* provést stažení všech JS, LESS a dalších závislostí.
3) Pomocí příkazu *gulp* dojde ke kompilaci všech závislostí a jejich umístění do *www* adresáře projektu.
4) Pomocí příkazu *composer install* dojde ke stažení a instalaci všech PHP knihoven do adresáře *vendor*.

8.5 Nasazení do produkčního prostředí

Vzhledem k tomu, že informační systém řídí celou počítačovou síť a při nasazení nového informačního systému došlo rovněž k obměně struktury počítačové sítě a jejího zabezpečení, nebylo

[38] NodeJS – Open-Source multiplatformní prostředí pro spouštění JavaScriptového kódu.
[39] Composer – nástroj pro správu knihoven a dalších PHP zdrojů.
[40] Gulp – nástroj pro automatizaci časově náročných a opakujících se úloh spojených s vývojem především webových aplikací, jako je např. minimalizace a spojování JavaScriptových a CSS kódů do větších souborů.

možné jednoduše nahradit původní informační systém novým, v opačném případě by totiž hrozilo odpojení členů klubu (a tedy ubytovaných na Hlávkově koleji) od sítě Internet. Proto bylo celé překlopení sítě a informačního systému dopředu precizně naplánováno a proběhlo v několika etapách.

8.5.1 Vytvoření nové počítačové sítě

Záložní (*slave*) server klubu byl kompletně přeinstalován a byl na něm nainstalován a nakonfigurován virtualizační software včetně všech kontejnerů s *master* a *slave* instancemi síťových služeb (DNS, Mail, DHCP, Radius, ad.) včetně nového informačního systému, do kterého byly přeneseny všechny uživatelské účty z původního řešení. Na tuto novou síťovou infrastrukturu pak byly napojeny Wi-Fi přístupové body a nový router. Celé toto řešení pak bylo připojeno do sítě Internet přes kaskádu dalších routerů, pomocí kterých pak byl nasimulován veřejný IP rozsah klubu (který v této chvíli stále využívalo původní řešení počítačové sítě).

8.5.2 Přesun členů na novou síť

Po zajištění dočasné konektivity byli vyzváni všichni členové klubu, aby se od 1. 3. 2017 do 31. 3. 2017 dostavili za správci sítě doplnit do nového informačního systému dodatečné osobní informace, které bylo podle nařízení Studentské unie ČVUT potřeba u členů od 1. 1. 2017 evidovat. Při této návštěvě byl zároveň každému členovi nastaven na všech zařízeních Wi-Fi profil, díky čemuž byl všem členům zajištěn nepřetržitý přístup k síti internet (byť rychlostně limitovaný).

8.5.3 Přechod na novou infrastrukturu

Od 1. 4. 2017 byla odpojena původní infrastruktura počítačové sítě. Během následujících 2 měsíců pak byl reinstalován hlavní (*master*) server, na který byly následně přesunuty všechny *master* instance síťových služeb a rekonfigurovány patrové switche. Dále byl pro členy sepsán detailní návod pro nastavení adaptérů kvůli nově zavedenému 802.1x zabezpečení na metalické konektivitě. V průběhu června 2017 pak došlo k aplikaci tohoto typu zabezpečení, čímž byly zároveň ukončeny všechny práce týkající se přechodu na nový informační systém.

8.6 Zhodnocení

Na základě předchozího návrhu byl implementován nový informační systém, který byl po bedlivé přípravě nasazen do produkčního prostředí tak, aby se během přechodu na nové řešení minimalizovaly jeho negativní vlivy a byla zajištěna nepřetržitá konektivita k síti Internet.

Dále byly představeny a rozebrány některé implementační detaily nového systému, rady pro jeho další vývoj a sepsán krátký návod pro sestavení (build) aplikace. V případě bližšího zájmu pak doporučuji nahlédnout přímo do zdrojových kódů aplikace. Samotný kód sice neobsahuje moc komentářů, nicméně díky pevně nastavenému programovacímu stylu v rámci celého projektu a jemnému členění funkcionality do menších funkčních celků o maximálně několika stovkách řádků kódu by se měl každý vývojář s běžnou zkušeností s Nette Frameworkem po krátkém přezkoumání kódu velmi rychle v implementaci zorientovat a být schopen funkcionalitu upravit podle svých potřeb.

9. Testování

Testování by mělo být nedílnou součástí každého vývoje software. Díky testování lze ověřit, že výsledný produkt splňuje všechny funkční a nefunkční požadavky, zjistit jeho požadavky na hardwarové a softwarové vybavení a v konečném důsledku tak i určit kvalitu výsledného produktu.

V této kapitole budou představeny provedené funkční a zátěžové testy. Na závěr pak bude provedeno zjednodušené testování paměťových a diskových nároků pro ověření, že produkční stroj, na kterém výsledná aplikace poběží, disponuje dostatečnými zdroji (hardware).

9.1 Funkční testování

Funkční testy slouží k ověření, že výsledná aplikace plní všechny úkoly, pro které je určena. Testuje se obecně veškerá funkcionalita implementovaná v aplikaci a ověřuje se, že tato funkcionalita pracuje správně, odpovídá požadavkům zákazníka a disponuje veškerou požadovanou funkcionalitou specifikovanou v požadavcích zákazníka.

9.1.1 Návrh testu

Pro testování funkčnosti aplikace budou implementovány Selenium testy [30]. Samotné testy pak budou implementovány v PHP pomocí Facebook Selenium WebDriveru [31] a Nette Testeru [32]. Pro testovací prostředí pak byl zvolen prohlížeč Chrome (za použití ChromeDriveru [33]) a rozlišení obrazovky 1920x1080.

Funkční testy budou pokrývat veškeré běžně vykonávané úkony, jako je:

- Přihlášení do aplikace.
- Zobrazení veškerých dostupných informací, které má uživatel k dispozici.
- Registrace nového člena (uživatele) do systému.
- Úprava osobních informací a základního nastavení uživatele.
- Přidání, úprava a odebrání síťových zařízení.
- Přidání, úprava a odebrání trestu u služby počítačová síť a fitcentrum
- Přidání, úprava a odebrání odpustku úhrady členského poplatku.
- Deaktivace a reaktivace členství a síťových služeb.

9.1.2 Výsledky testu

Funkční testy odhalily v aplikaci tyto chyby:

1) Při odeslání formuláře pro úpravu osobních informací, který obsahoval změnu jména a příjmení, přes AJAX, nedošlo k propagaci této změny do horní navigace aplikace. Tato chyba byla způsobena opomenutím aktualizace této komponenty v odpovědi aplikace.
2) Při registraci trestu s okamžitou platností a při zrušení právě účinného trestu u služby nedošlo k propagaci této změny do komponenty zobrazující stav služeb (aktivní, neaktivní, trest). Chyba byla opět způsobena opomenutou aktualizací této komponenty při AJAX dotazu.
3) Uživatelům, kteří měli přiřazenou pouze roli *člen rady,* byl odepřen přístup na stránku s výpisem členů klubu. Tato chyba byla způsobena absencí této role v seznamu rolí, které mají k této stránce přístup v presenteru této stránky.
4) *Jiní uživatelé, než administrátor, registrátor, správce sítě a správce posilovny* nemohli měnit svůj email a telefon. Chyba byla zapříčiněna inicializací formuláře, ve kterém se tyto vstupní prvky nejprve vytvořili jako *disabled* a až těsně před vykreslením se aktivovaly, tj. při validaci formuláře byly tyto prvky zakázány a formulář tak vracel *null* hodnotu.
5) Uživatelé, kteří měli přiřazenou pouze roli p*ředseda,* nemohli udělovat členům odpustky. Chyba byla zapříčiněna absencí této role v seznamu rolí, které mají k této funkcionalitě přístup.

9.1.3 Zhodnocení testu

Pomocí funkčního testování byly objeveny chyby v implementaci aplikace, které byly následně opraveny. Testováním byla zároveň ověřena správnost implementace aplikace a splnění definovaných funkčních požadavků systému.

9.2 Zátěžové testování aplikace

Dalším důležitým kritériem na posouzení kvality webové aplikace je doba odezvy, tj. uplynulý čas od okamžiku odeslání požadavku do přijetí odpovědi, přičemž platí, že menší doba odezvy znamená výkonnější aplikaci.

Při krátké době odezvy má uživatel celkově dojem plynulejší aplikace a může tak rychle a efektivně pracovat. Naopak dlouhá doba odezvy může u uživatele způsobit dojem „zamrzávání“ aplikace, což pak může vést i k tomu, že uživatel požadavek kvůli netrpělivosti odešle vícekrát, v důsledku čehož čeká na odpověď ještě déle. Opakované odeslání požadavku pak může způsobit i neočekávané chování aplikace, kdy například při smazání záznamu z databáze se příkaz sice napoprvé provede správně, ale uživatel uvidí až odpověď posledního požadavku, který selhal, jelikož záznam byl již smazán dřívějším požadavkem.

Cílem je tak udržovat dobu odezvy co nejmenší, a to ideálně do několika stovek milisekund. Jako ještě únosná bude uvažována odezva do 1 sekundy. Odezva nad 1 sekundu pak bude považována za nedostačující.

Dalším kritériem pro určení výkonu aplikace je množství požadavků, které může aplikace obsloužit za daný čas. Toto číslo pak většinou reprezentuje, kolik uživatelů může aplikaci najednou používat, přičemž zde naopak platí, že větší hodnota znamená výkonnější aplikaci (více uživatelů).

9.2.1 Návrh testu

Pro zátěžové testování aplikace bude použit JMeter [34]. Jedná se o Open-Source aplikaci napsanou v jazyce Java, která slouží pro funkční testování chování aplikace při zátěži a měření výkonu.

Při testování nás bude zajímat počet simulovaných uživatelů, průměrná doba odezvy na požadavek a počet zpracovaných požadavků za 1 sekundu. Testovat budeme chování při 10, 20 a 50 uživatelích (vláknech), přičemž vždy jedna polovina uživatelů bude dělat *GET* dotazy (pro zobrazení stránky a získání dat) a druhá polovina uživatelů *POST* dotazy (vytvářet a upravovat data).

Důvodem rozdělení dotazů při testování na *GET* a *POST* je předpoklad, že *POST* dotazy budou trvat déle, jelikož při každém takovémto dotazu je potřeba znovu navázat spojení s OpenLDAP databází a uložit do ní nová data. Dalším důvodem rozdělení je, že počet *GET* dotazů bude při běžném používání aplikace větší, protože uživatel (administrátor) si musí nejprve zobrazit výpis uživatelů, otevřít profil uživatele, kterého chce upravovat, a až následně může editovat jeho nastavení. Navíc se na některých stránkách načítají data asynchronně přes AJAX pro zrychlení práce na stránce a pohodlnější ovládání. Proto jsou testy navrženy tak, aby byl poměr *GET* a *POST* dotazů alespoň 2:1.

Testování bude probíhat na produkčním serveru, tj. v reálných podmínkách a na hardware, který je popsán v nefunkčních požadavcích v kapitole 5.1.

9.2.2 Testovací plán

Testovací plán pro JMeter je přiložen na CD ve formátu *JMX*.

9.2.2.1 GET dotazy

- Zobrazení přihlašovací stránky
- Přihlášení se do aplikace
- Zobrazení profilu uživatele
- Zobrazení nastavení sítě
- Zobrazení nastavení fitcentra

- Získání údajů o vstupech do fitcentra
- Zobrazení stránky s členskými příspěvky
- Zobrazení informací o členském příspěvku
- Zobrazení stránky se členy (administrátor)
- Získání seznamu členů (administrátor)
- Zobrazení stránky s bankovními transakcemi (administrátor)
- Získání seznamu bankovních transakcí (administrátor)
- Zobrazení detailu bankovní transakce (administrátor)
- Zobrazení stránky s historií přiřazených VLAN sítí (administrátor)
- Získání seznamu přiřazených VLAN sítí (administrátor)
- Zobrazení stránky pro odesílání emailů (administrátor)
- Zobrazení stránky s nastavením aplikace (administrátor)

9.2.2.2 POST dotazy

- Přihlášení se do aplikace
- Aktualizace osobních údajů
- Aktualizace přístupových údajů
- Přidání zařízení
- Přidání síťového trestu
- Zaznamenání vstupu do fitcentra
- Přidání trestu ve fitcentru
- Vytvoření nového člena (uživatele)

9.2.3 Výsledky testování

Při 10 uživatelích (5 *GET*, 5 *POST*) byla aplikace schopná obsloužit 2‘127 požadavků za 1 minutu, tj. 3,55 požadavků na uživatele za 1 sekundu při průměrné době odezvy 201 milisekund. Výstup z měření demonstruje Obrázek 7 v příloze.

Při 20 uživatelích (10 *GET*, 10 *POST*) byla aplikace schopná obsloužit 2‘684 požadavků za 1 minutu, tj. 2,24 požadavků na uživatele za 1 sekundu při průměrné době odezvy 361 milisekund. Výstup z měření demonstruje Obrázek 8 v příloze.

Při 50 uživatelích (25 *GET*, 25 *POST*) byla aplikace schopná obsloužit 2‘583 požadavků za 1 minutu, tj. 0,86 požadavků na uživatele za 1 sekundu při průměrné době odezvy 1‘026 milisekund. Výstup z měření demonstruje Obrázek 9 v příloze.

9.2.4 Zhodnocení testu

Zátěžovým testem bylo zjištěno, že je aplikace schopná obsloužit průměrně 35 dotazů za 1 sekundu s průměrnou dobou odezvy 201 milisekund / dotaz, čímž byly splněny nefunkční požadavky na výkon aplikace z kapitoly 5.3.

Maximální počet dotazů, které je aplikace průměrně schopna obsloužit se pak pohybuje kolem 40 dotazů za 1 sekundu, přičemž při větším počtu dotazů se začne adekvátně zvětšovat doba odezvy. Je ovšem potřeba si uvědomit, že v reálných podmínkách udělá uživatel v průměru jen cca 1 dotaz za 1 sekundu. Aplikace by tak měla být ve výsledku schopná obsloužit i více jak 40 intenzivně pracujících uživatelů.

Dále byla při testech potvrzena domněnka, že *POST* dotazy budou oproti *GET* dotazům časové náročnější (viz grafy přiložené na CD). V průměru pak byly *POST* dotazy časově 2,6krát náročnější, což je dáno jednak povahou dotazů (při *GET* dotazu většinou pouze získáváme data z MySQL databáze, při *POST* dotazu zpracováváme vstupní data, provádíme kontroly atd., v důsledku čehož je výsledná režie větší) a jednak již zmíněnou OpenLDAP databází. Důvodem je, že používaná PHP knihovna pro komunikaci s OpenLDAP databází si na rozdíl od používané PHP knihovny MySQL neuchovává připojení k databázi mezi jednotlivými dotazy (http požadavky). Při každém *POST* dotazu, který propisuje data do OpenLDAP databáze, se tak musí připojení navázat znovu, což stojí značný čas a zvětšuje se tak celková odezva aplikace. V případě problémů s výkonem by tak jedním z řešení mohlo být rozšíření konektoru OpenLDAP knihovny, které by pro každého uživatele vytvořilo

právě jedno připojení k OpenLDAP databázi a mezi http dotazy jej ukládalo do *CACHE* paměti přiřazené *SESSION* relaci uživatele (stejně jako tomu u MySQL knihovny).

9.3 Testování paměťových nároků

Motivací pro tento test bylo zjistit paměťové nároky aplikace, a zdali její případné navýšení, případně zvýšení hodnoty *memory_limit* v konfiguraci PHP z defaultních 128 MB, umožní obsloužit více požadavků za daný časový úsek (zvýší výkon aplikace).

9.3.1 Návrh testu

Pro simulaci zátěže bude použit stejný testovací plán jako u zátěžového testování v předchozí kapitole 9.2. Dále bude použit program *top* [35], což je interaktivní terminálová aplikace sloužící k zobrazení běžících procesů a jejich využívaných zdrojů na Linux OS (obdoba Správce úloh z Windows). Zajímat nás bude především počet běžících procesů obsluhujících dotazy aplikace a jejich využití paměti.

9.3.2 Výsledky testování

Při intenzivní zátěži a simulování 50 aktivních uživatelů se počet procesů pohyboval okolo 20 až 25, přičemž každý z procesů využíval od 4 do 7 MB paměti. Maximální využitá paměť webovým serverem pak byla do 280 MB. Paměť využitá systémem se pak pohybovala okolo 354 MB a velikost volné paměti se tak pohybovala okolo 1,4 GB.

9.3.3 Zhodnocení testu

Při testování bylo zjištěno, že paměť stroje je víc než dostatečně naddimenzována a s největší pravděpodobností by stačilo stroji přiřadit jen 1 GB paměti se *swap* oddílem o poloviční velikosti, aniž by tím byl výrazně ovlivněn výkon informačního systému.

9.4 Testování nároků na velikost disku

Motivací pro tento test byly zkušenosti z praxe, kdy se nejednou stalo, že server, na kterém aplikace běžela, zamrznul z důvodu nedostatku diskového prostoru. Proto je dobrou praxí položit si otázku, jak bude aplikace využívána, jaký lze očekávat nárůst dat v databázi za dané období (obvykle rok) a podle těchto předpokladů pak odhadnout, kolik zdrojů bude výsledná aplikace ke svému běhu vyžadovat.

U této aplikace se sice nepředpokládají závratné nároky na diskový prostor, nicméně z důvodu povahy aplikace je vhodné pravidelně vytvářet zálohy databáze, a to ideálně na denní bázi, a ukládat je jednak na disk serveru (aby byly snadno dostupné), jednak také nahrát kopii na vzdálené úložiště (např. přes FTP). Navíc bude pro každého nového člena generován přihlašovací formulář s jeho osobními informacemi v PDF. Je proto dobrým nápadem zkusit odhadnout nároky na prostředky těchto úloh a v návaznosti na zjištění stanovit, jak často bude potřeba tyto soubory promazávat a za jak dlouhé období bude možné držet zpětné zálohy databáze.

9.4.1 Předpoklady

Při průzkumu databáze původního systému bylo zjištěno, že během 1 roku se na koleji ubytuje zhruba 60 až 80 nových studentů. Pro účely testu proto budeme uvažovat o založení 100 nových členských účtů a o deaktivaci 100 starých účtů od odchozích členů.

U každého uživatele jsou pak v průměru registrovány 4 MAC adresy (Ethernet a Wi-Fi adresa počítače a Wi-Fi adresa telefonu a tabletu).

Počet aktivních členů se pohybuje okolo 200, díky čemuž je ročně přijato 400 členských příspěvků a 20 % členů využívá pravidelně fitcentrum, a to zhruba 2 až 3krát do týdne, což odpovídá cca 2‘000 registrovaných vstupů do fitcentra ročně. Pro účely testu se bude předpokládat dvojnásobný počet registrovaných vstupů.

Rovněž bylo zjištěno, že zhruba 25 % členů neuhradí včas členský příspěvek, což vede k dočasné deaktivaci jejich účtu. Pro účely testu bude předpokládána dočasná deaktivace 50 % všech účtů.

9.4.2 Návrh testu

Pro účely testu bude použita aplikace JMeter, která bude simulovat úkony, které jsou během 1 roku provedeny v aplikaci. Simulace bude spuštěná několikrát, přičemž bude sledován nárůst velikosti zálohy databáze.

Testovací plán simulující 1 rok provozu bude zahrnovat:

- Založení 100 nových členů.
- Aktualizace osobních informací u 100 členů.
- Registrace 400 MAC adres.
- Zaznamenání 4‘000 vstupů do fitcentra.
- Zaznamenání 400 příchozích plateb za členské příspěvky.
- Udělení 100 trestů u počítačové sítě.
- Udělení 100 trestů u fitcentra.
- Deaktivace 200 uživatelských účtů.
- Reaktivace 100 uživatelských účtů.

9.4.3 Výsledky testování

Výsledky měření ukázaly, že meziroční nárůst zálohy se při výše uvedených předpokladech pohybuje okolo 150kB. Velikost generovaných PDF souborů s registračním formulářem se pak pohybuje okolo 176 až 178kB.

Pro následující výpočty budeme uvažovat, že stávající velikost zálohy databáze produkčního serveru je 400 kB. Dále budeme uvažovat, že byť se velikost zálohy mění hlavně na přelomu semestru (skokově), lze z dlouhodobého hlediska (roky) uvažovat lineární nárůst velikosti zálohy, a tedy i spotřebovaného místa úložiště. Odhad potřebného diskového místa pak bude dán vzorcem:

$$S = \frac{d * n}{2} + v * n$$

kde d představuje velikost meziročního nárůstu zálohy, v stávající velikost zálohy (400kB) a n celkový počet záloh za 10 let (3‘652).

V případě optimistického odhadu, kdy velikost meziročního nárůstu nepřesáhne 200kB tak lze očekávat, že potřebný diskový prostor pro ukládání záloh databáze za období 10 let bude zhruba 4,88 GB. V případě pesimistického odhadu, kdy velikost meziročního nárůstu nepřesáhne 500kB pak lze očekávat, že potřebný diskový prostor pro ukládání záloh databáze za období 10 let bude zhruba 10,1 GB.

Co se týče registračních formulářů, tak zde je výpočet jednodušší. Během 10 let se registruje zhruba 1‘000 nových členů. Při velikosti jednoho formuláře 178kB tak lze očekávat nároky na diskový prostor o velikosti zhruba 178MB.

Nároky na databázi jsou pak zanedbatelné a měly by se pohybovat v řádu desítek, maximálně pak několika stovek MB.

9.4.4 Zhodnocení testu

Testováním a následným expertním odhadem bylo zjištěno, že i při pesimistickém odhadu, kdy je předpokládáno rozšíření stávající struktury databáze a ukládání většího množství informací, by systém neměl vyžadovat více jak 10 až 11GB diskového prostoru za dobu 10 let nepřetržitého provozu.

Testem tak bylo dokázáno, že kapacita diskového úložiště o velikosti 32 GB (viz specifikace hardware v nefunkčních požadavcích, kapitola 5.1) je pro potřeby informačního sytému více než dostačující.

9.5 Zhodnocení

Funkční testování aplikace probíhalo pomocí předem připravených Selenium testů. Během testování se podařilo odladit několik drobných chyb týkajících se především opomenuté aktualizace částí grafického rozhraní při ukládání změn (v důsledku zpracování požadavků na pozadí prohlížeče pomocí technologie AJAX) a chybně nastavených ACL oprávnění. Závaznější chyby se nepodařilo odhalit a aplikace tak byla nasazena do produkčního prostředí, kde prozatím funguje bez dalších zjištěných chyb.

Dále bylo provedeno zátěžové testování aplikace, přičemž bylo zjištěno, že server specifikovaný v nefunkčních požadavcích je schopen obsloužit 40 i více intenzivně pracujících uživatelů s dobou odezvy do 500 milisekund. V reálném prostředí se pak předpokládají maximálně 2 souběžně operující uživatelé, přičemž doba odezvy při takovém to zatížení by měla být do 200 milisekund.

Na závěr bylo provedeno testování paměťových a diskových nároků aplikace pro ověření, že produkční server disponuje dostatečnými prostředky pro dlouhodobý nepřetržitý běh informačního systému i při vysoké zátěži. Provedenými testy bylo následně zjištěno, že specifikovaný hardware je značně naddimenzovaný a pro předpokládané využívání by mělo být dostačujících i 1GB RAM paměti a 8GB diskového prostoru.

10. Závěr

Hlavním cílem této práce bylo odstranit nedostatky původního řešení informačního systému pro správu členů a služeb klubu Hlávkova kolej, které bylo značně zastaralé, neintuitivní, chybové a disponovalo značně omezenou funkcionalitou. Další motivací k této práci pak bylo rozšíření původního řešení o další funkcionalitu týkající se především lepšího členění uživatelských rolí, možnost evidovat u uživatelů a zařízení poznámky a propracovanější kontrolu nad využívanými službami členů.

Vzhledem k těmto požadavkům byla provedena hloubková analýza původního řešení informačního systému, jejímž výsledkem byla sumarizace veškeré dostupné funkcionality a jejích nedostatků. Spolu s analýzou původního informačního systému byla provedena i analýza řešení počítačové sítě a jejich požadavků, které byly kladeny na původní informační systém. Během této analýzy pak byly odhaleny zásadní nedostatky v návrhu počítačové sítě, které se týkaly především zabezpečení počítačové sítě, absence redundance a nemožnosti registrovat u uživatele více jak jedno zařízení. Nad rámec této práce proto bylo navrženo a následně implementováno nové řešení infrastruktury počítačové sítě, které tyto nedostatky odstranilo, a byla představena analýza nového řešení spolu s jeho požadavky na nový informační systém.

Na základě zjištěných výsledků analýzy původního řešení, nových požadavků klubu Hlávkova kolej a požadavků nové infrastruktury počítačové sítě byly navrženy a představeny ucelené funkční a nefunkční požadavky na nový informační systém. S ohledem na tyto požadavky pak byla provedena analýza dostupných existujících řešení v podobě jiných kolejních informačních systémů Studentské unie ČVUT. Touto analýzou pak bylo zjištěno, že tato řešení jsou buď velmi zastaralá, nebo vyžadují vyšší odbornou znalost pro jejich vývoj a údržbu, kvůli čemuž jsou nevhodné pro potřeby klubu Hlávkova kolej.

Vzhledem k těmto zjištěním pak byl na základě sepsaných požadavků navrhnut a implementován nový informační systém nad technologií PHP, HTML a JavaScript. Tento systém pak byl podroben funkčním a zátěžovým testům pro ověření splnění funkčních a nefunkčních požadavků. Současně s testy pak byly provedeny a odhadnuty přibližné požadavky implementovaného řešení na potřebný hardware.

Během analýzy existujících řešení bylo zjištěno, že klub Orlík v současné době nedisponuje vlastním informačním systémem. Jelikož má tento klub téměř totožné požadavky na informační systém jako klub Hlávkova kolej, byl výsledný systém tomuto klubu nabídnut a k jeho nasazení, by mělo dojít kolem února až března 2018.

10.1 Průběh práce

Prvním impulzem k této práci byla žádost klubu Hlávkova kolej o úpravu implementace jejich informačního systému a především možnost pokrytí celé Hlávkovy koleje Wi-Fi konektivitou.

Na začátku roku 2016 jsem proto provedl hloubkovou analýzu původního řešení s ohledem na tyto požadavky, na jehož základě jsem pak představil možné alternativy řešení. Během letních prázdnin roku 2016 jsem pak navrhl a implementoval nové řešení počítačové sítě a pokryl celou kolej Aruba Wi-Fi AP, které jsem pak následně testoval až do podzimu 2016.

V zimě 2016 pak byly zahájeny první práce na novém informačním systému a koncem ledna 2017 byla představena první demoverze nového systému, která obsahovala základní nejnutnější funkcionalitu potřebnou k zajištění provozu klubu a počítačové sítě. Následně byla tato demoverze po dobu 1 měsíce testována v testovacím prostředí. Po ujištění se o její funkčnosti byli členové během následujících 2 měsíců systematicky přesouváni na nové řešení tak, aby nebyl omezen jejich přístup k síti Internet. Po ukončení tohoto období pak bylo odstaveno původní řešení a celá síť uvedena do podoby, která je popsána v kapitole 3.

Samotný informační systém pak byl dále rozšiřován o další dodatečnou funkcionalitu popsanou ve funkčních požadavcích až do konce srpna 2017, kdy byly ukončeny veškeré práce na informačním systému popisované v této práci.

10.2 Zhodnocení přínosu

Zhodnocení bude provedeno z pohledu běžného uživatele (člena klubu Hlávkova kolej) a správce systému (administrátor, člen rady, správce služby).

10.2.1 Běžný uživatel

Co se týče grafického rozhraní aplikace, tak jeho návrh je podstatně přívětivější a intuitivnější. Uživatel se může velmi snadno dostat ke všem svým dostupným informacím bez potřeby složitého problikávání a prozkoumávání dostupné funkcionality, jako tomu bylo u původního řešení, které kopírovalo grafické rozhraní operačního systému, tj. jednotlivá funkcionalita byla dostupná po „rozkliknutí“ ikonek na ploše nebo v nabídce „Start“.

Dalším velkým přínosem pak určitě je responzivní design aplikace a její celá lokalizace do českého a anglického jazyka (původní systém podporoval pouze češtinu) a to včetně automaticky rozesílaných emailových notifikací, což velmi ocenili hlavně zahraniční studenti.

Systém rovněž nabízí lepší přehled nad úhradou členských poplatků, uživatele včasně upomíná o jejich úhradě, díky čemuž klesl počet opožděných plateb o více jak 50 %.

V poslední řadě pak členové ocenili urychlení některých procesů, konkrétně pak registrace příchozích úhrad členských poplatků, registrace do klubu a registrace nových zařízení do počítačové sítě, do které získávají přístup v okamžiku registrace síťového zařízení, tj. bez několikaminutového čekání na vygenerování a propagaci nového nastavení na síťové prvky.

10.2.2 Správce

Kromě přínosu, který je uveden u běžného uživatele, se pak jedná o celkově lepší přehled nad správou členů a jejich služeb, možnost členovi odsunout úhradu členského příspěvku a vedení evidence historie využívání služeb.

Rozhraní systému je uživatelsky přívětivé a systém zcela autonomně provádí většinu úloh spojených s administrací (přiřazuje VLAN sítě, deaktivuje a reaktivuje uživatelské účty a služby v závislosti na přijatých platbách a trestech vedených u služeb ad.). Rovněž odpadla potřeba dodržovat přesné postupy, např. při odpojení zařízení nebo uživatele od počítačové sítě, a téměř veškeré nastavení se propaguje na služby okamžitě, díky čemuž lze snadno a rychle dohledat např. případný problém s připojením počítače do sítě, bez potřeby několikaminutového čekání.

Jako poslední (mimo další) bych pak chtěl zmínit napojené emailové aliasy na přiřazené systémové role a implementovaný přehled výpisu bankovního účtu, který je dostupný i všem členům kolejní rady.

10.3 Návrhy na další vylepšení

Návrhů na další vylepšení a rozšíření systémů je plno, proto zde uvedu pouze několik nejzajímavějších.

10.3.1 Lokalizace do dalších jazyků

Vzhledem k tomu, že na Hlávkově koleji a koleji Orlík se v současné době nachází velké množství rusky mluvících studentů, stojí za uvážení případné rozšíření informačního systému o ruskou lokalizaci. Osobně bych pak ale chtěl doporučit, aby se informační systém nepřekládal do značného množství jazyků, jelikož by to mohlo mít negativní vliv na možnost implementace další funkcionality (jelikož každou novou funkcionalitu by pak bylo potřeba překládat do všech těchto jazyků).

10.3.2 Úprava ACL

V návrhu systému jsem nepředpokládal potřebu pokročilejšího autorizačního systému a pro zjednodušení práce autorizuji uživatele pouze na základě přiřazených rolí. Postupem času se ale ukázalo, že toto řešení není praktické, protože vyvstala potřeba do systému přidat novou systémovou roli *správce deskových her,* v důsledku čehož pak bylo potřeba upravit kód systému tak, aby této roli byla dostupná některá další funkcionalita.

Vzhledem k tomuto zjištění tak bude v dohledné době systém upraven a přístup k funkcionalitě bude řízen na základě povolených akcí u jednotlivých rolí.

10.3.3 Předregistrace

Pro urychlení registrace nových uživatelů by systém mohl implementovat veřejný předregistrační formulář, který by zájemce o vstup do klubu vyplnil před návštěvou registrátora. Registrátor by pak pouze zkontroloval správnost uvedených informací a schválil založení účtu, díky čemuž by pak mohl rychleji obsloužit více nových členů.

10.3.4 Přístupy pro administrátory

Na všechny servery klubu byla nasazena PAM autentifikace proti LDAP databázi. Pokud je tak uživateli v informačním systému přidělena role *administrátor* získává automaticky SSH přístup na všechny tyto stroje. Pro zvýšení bezpečnosti tohoto řešení by pak mohl informační systém umožnit správu veřejných SSH klíčů, které by pak prostřednictvím LDAP databáze propagoval na jednotlivé servery.

Dále byly vytvořeny dva OpenVPN servery pro vzdálený přístup do servisní sítě. K těmto serverům je ale potřeba manuálně generovat pro administrátory osobní certifikáty. Informační systém by tak mohl být rozšířen i o možnost vygenerovat si v něm důvěryhodné certifikáty pro připojení do VPN sítě.

10.3.5 GDPR

V květnu 2017 vstoupí v platnost nová regulace o ochraně osobních dat (GDPR), která zpřísňuje podmínky manipulace s osobními daty. V dohledné době tak bude potřeba konzultovat tuto normu s právníkem a upravit informační systém podle potřeby.

Minimálně pak bude nutné implementovat funkcionalitu pro jednoduché smazání, případně kompletní anonymizaci členů, kteří budou vyžadovat zrušení souhlasu o zpracování osobních dat. Dále bude potřeba implementovat mechanizmy pro automatickou anonymizaci členů po uplynutí maximální doby, pro kterou dal člen souhlas o uchování jeho osobních informací.

Zdroje

[1] Historie. *Studentská unie ČVUT* [online]. verze Červen 2009 [cit. 2016-01-05]. Dostupné z: http://su.cvut.cz/cs/historie

[2] DRAGOUNOVÁ, A. *Administrativní rozhraní počítačové sítě*. Praha: 2011. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra počítačů. Dostupné také z: https://dip.felk.cvut.cz/browse/details.php?f=F3&d=K13136&y=2011&a=dragoade&t=bach

[3] PERRY, J. M. *ExtJS*. Oreilly & Assoc Inc, 2014. ISBN-10: 1449361978.

[4] Zdroják.cz. *Bezpečnost na webu – přehled útoků na webové aplikace* [online]. 10. 11. 2008 [cit. 2017-03-15]. Dostupné z: https://www.zdrojak.cz/clanky/prehled-utoku-na-webove-aplikace/

[5] GOLDMAN, R. *Learning Proxmox VE*. Packt Publishing, 2016. ISBN-10: 1783981784.

[6] O informačním systému klubu Silicon Hill. *IS.sh* [online]. [cit. 2017-04-16]. Dostupné z: https://is.sh.cvut.cz/about

[7] *Bootstrap* [online]. [cit. 2014-05-12]. Dostupné z: http://getbootstrap.com

[8] MELE, A. *Django By Example*. Packt Publishing, 2015. ISBN-10: 1784391913.

[9] BOSCHI, S. *RabbitMQ Cookbook*. Packt Publishing, 2013. ISBN-10: 1849516502.

[10] Smarty. *Smarty - PHP Template Engine* [online]. [cit. 2017-05-16]. Dostupné z: https://www.smarty.net/

[11] Nette Framework [online]. [cit. 2017-11-09]. Dostupné z: https://nette.org/cs/

[12] ALMSAEED, A. AdminLTE [online]. [cit. 2016-10-12]. Dostupné z: https://adminlte.io/

[13] DUCKETT, J. *JavaScript and JQuery: Interactive Front-End Web Development*. John Wiley & Sons, 2014. ISBN-10: 1118531647.

[14] DataTables [online]. [cit. 2016-10-12]. Dostupné z: https://datatables.net/

[15] Input Mask [online]. [cit. 2016-10-12]. Dostupné z: https://github.com/RobinHerbots/Inputmask

[16] Github. *Are You Sure* [online]. [cit. 2016-10-13]. Dostupné z: https://github.com/codedance/jquery.AreYouSure

[17] Github. *Toastr* [online]. [cit. 2016-10-13]. Dostupné z: https://github.com/CodeSeven/toastr

[18] Github. *Kdyby/Translation* [online]. [cit. 2016-10-13]. Dostupné z: https://github.com/Kdyby/Translation

[19] Symfony [online]. [cit. 2016-10-13]. Dostupné z: https://symfony.com/

[20] Github. *Kdyby/Console* [online]. [cit. 2016-10-13]. Dostupné z: https://github.com/Kdyby/Console

[21] Github. *Kdyby/Events* [online]. [cit. 2016-10-13]. Dostupné z: https://github.com/Kdyby/Events

[22] Github. *Kdyby/Doctrine* [online]. [cit. 2016-10-13]. Dostupné z: https://github.com/Kdyby/Doctrine

[23] DUNGLAS, K. *Persistence in PHP with Doctrine ORM*. Packt Publishing, 2013. ISBN-10: 1782164103.

[24] Github. *Zenify/Doctrine-migrations* [online]. [cit. 2016-10-13]. Dostupné z: https://github.com/DeprecatedPackages/DoctrineMigrations

[25] Github. *Zenify/Doctrine-behaviours* [online]. [cit. 2016-10-13]. Dostupné z: https://github.com/DeprecatedPackages/DoctrineBehaviors

[26] Github. *Doctrine 2 Extensions* [online]. [cit. 2016-10-13]. Dostupné z: https://github.com/wiredmedia/doctrine-extensions

[27] Github. *TIESA Ldap Component* [online]. [cit. 2016-10-13]. Dostupné z: https://github.com/ccottet/ldap

[28] TCPDF [online]. [cit. 2016-10-13]. Dostupné z: https://tcpdf.org/

[29] Nette Framework. *Přihlašování & oprávnění uživatelů* [online]. [cit. 2016-10-13]. Dostupné z: https://doc.nette.org/cs/2.4/access-control

[30] PRASAD, R. *Learning Selenium Testing Tools*. Third Edition. Packt Publishing, 2015. ISBN-10: 1784396494.

[31] GitHub - facebook/php-webdriver: A php client for webdriver. [online]. [cit. 2017-05-12]. Dostupné z: https://github.com/facebook/php-webdriver

[32] Nette Tester - pohodové testování | Nette Framework [online]. [cit. 2017-05-12]. Dostupné z: https://tester.nette.org/cs/

[33] ChromeDriver - WebDriver for Chrome [online]. [cit. 2017-05-12]. Dostupné z: https://sites.google.com/a/chromium.org/chromedriver/

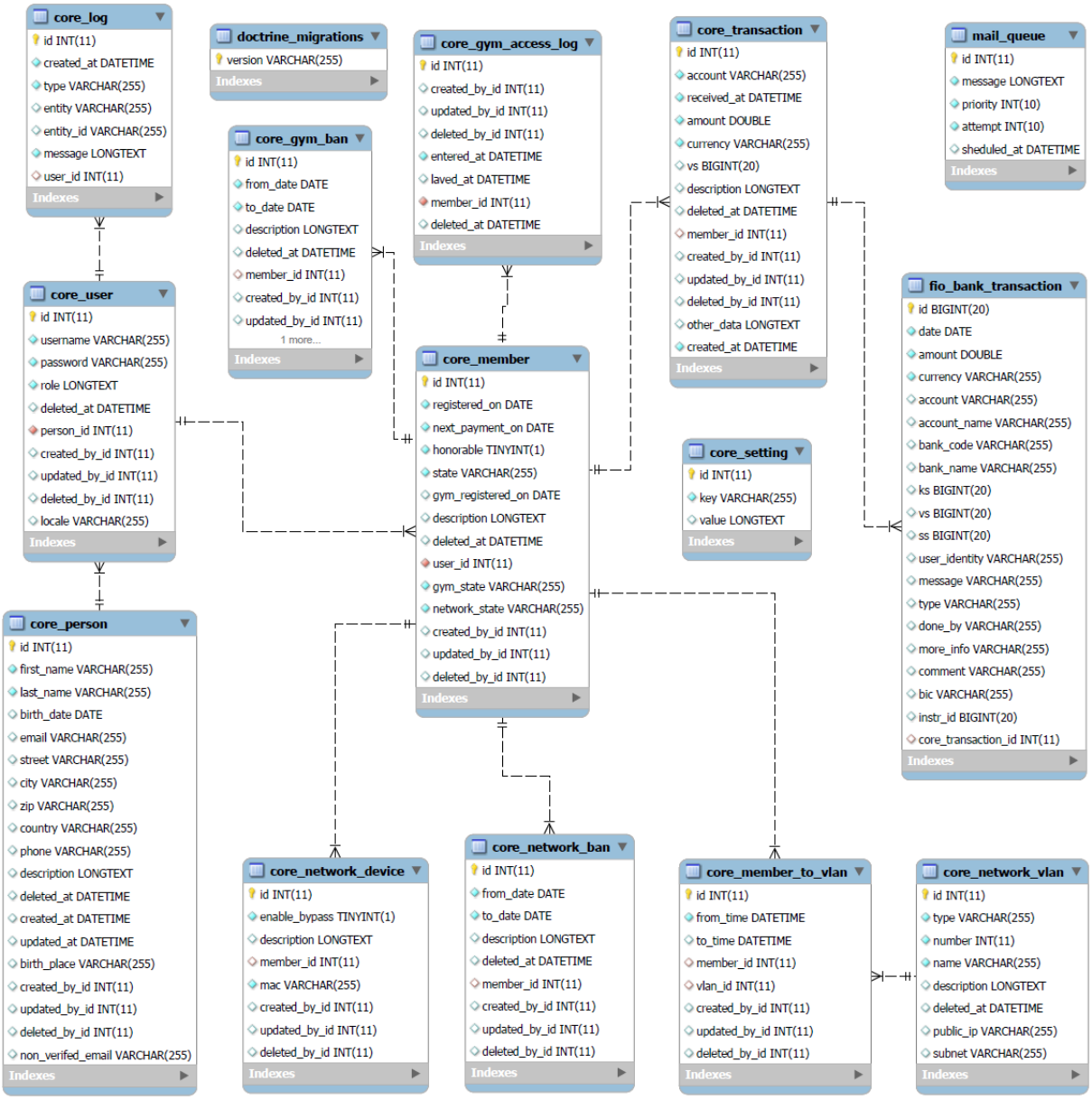
[34] HALILI, E. H. *Apache JMeter: A practical beginner's guide to automated testing and performance measurement for …*. Packt Publishing, 2008. ISBN-10: 1847192955.

[35] Linux manual page. *top* [online]. [cit. 2017-12-15]. Dostupné z: http://man7.org/linux/man-pages/man1/top.1.html

Seznam obrázků

Příloha 1

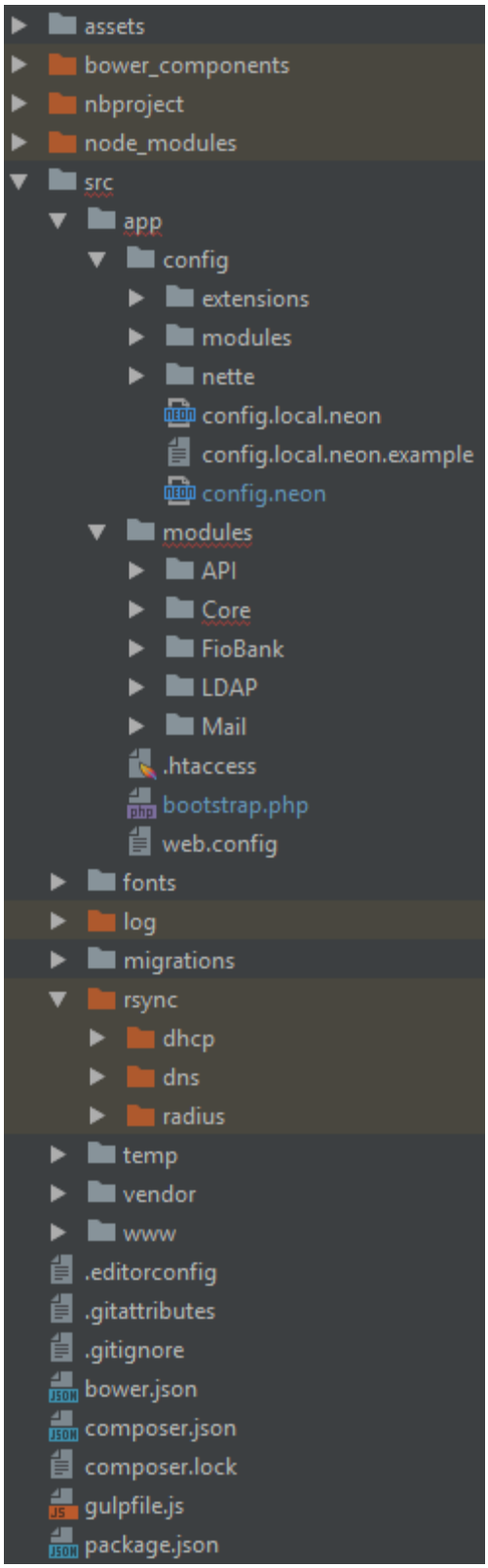
Návrh schéma databáze



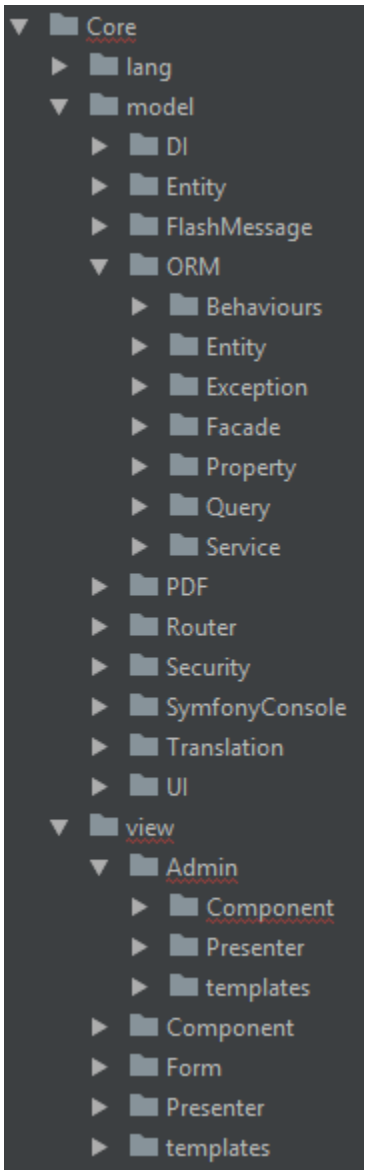
Obrázek 4 - Návrh schéma databáze

Příloha 2

Struktura projektu



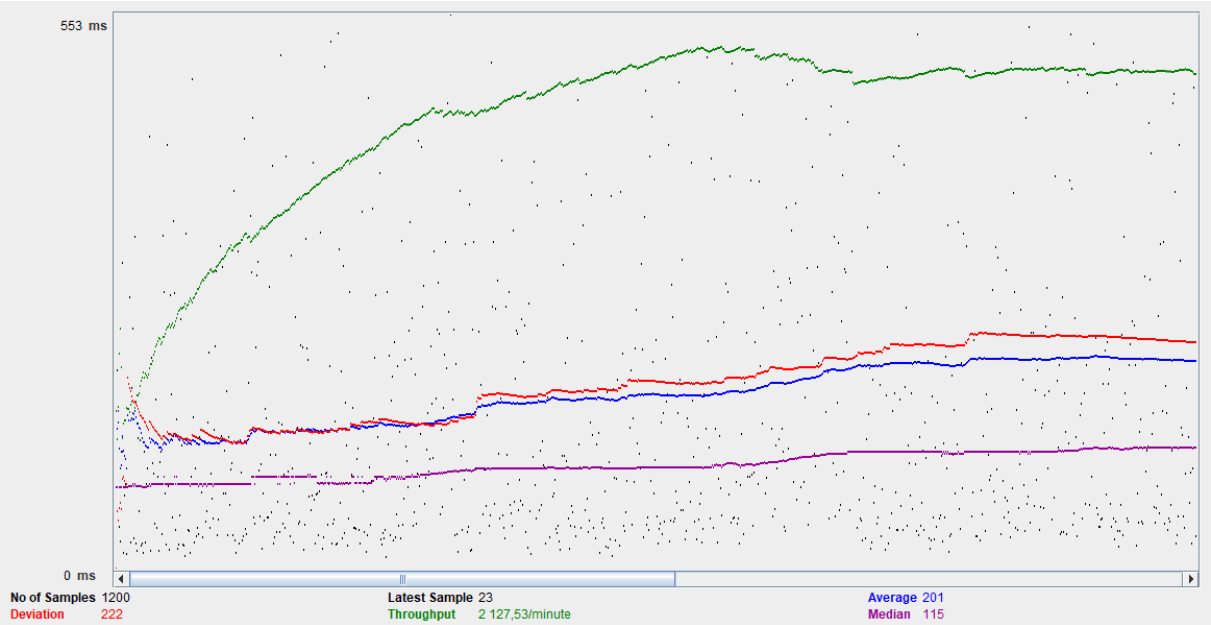
Obrázek 5 - Struktura projektu



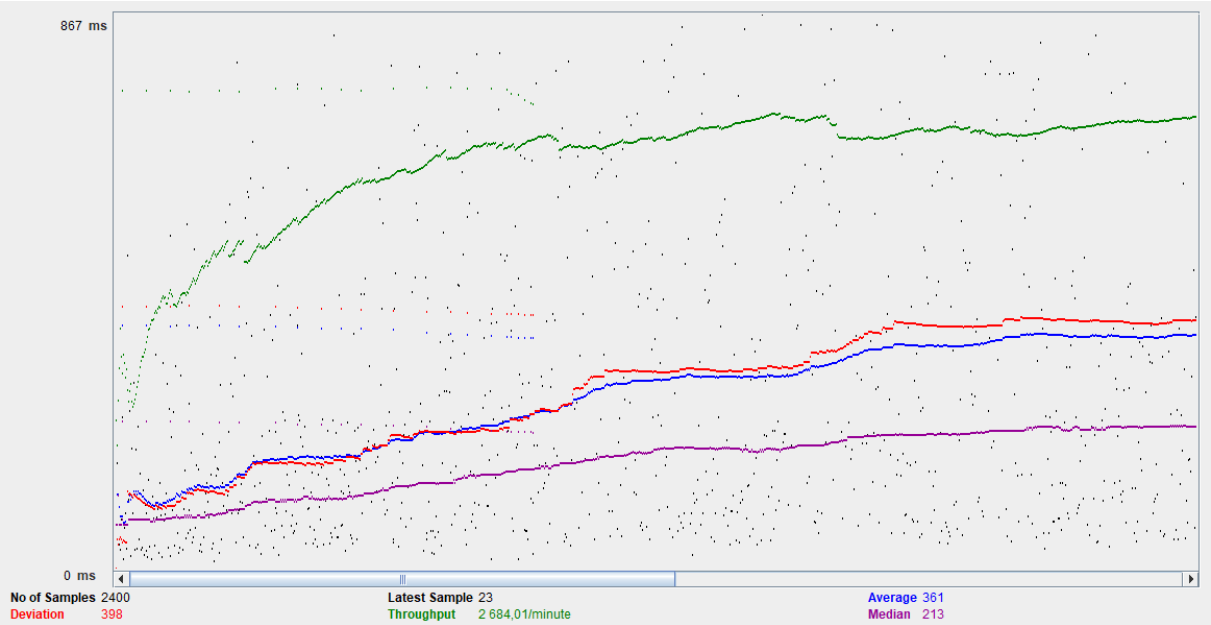
Obrázek 6 - Struktura modulu

Příloha 3

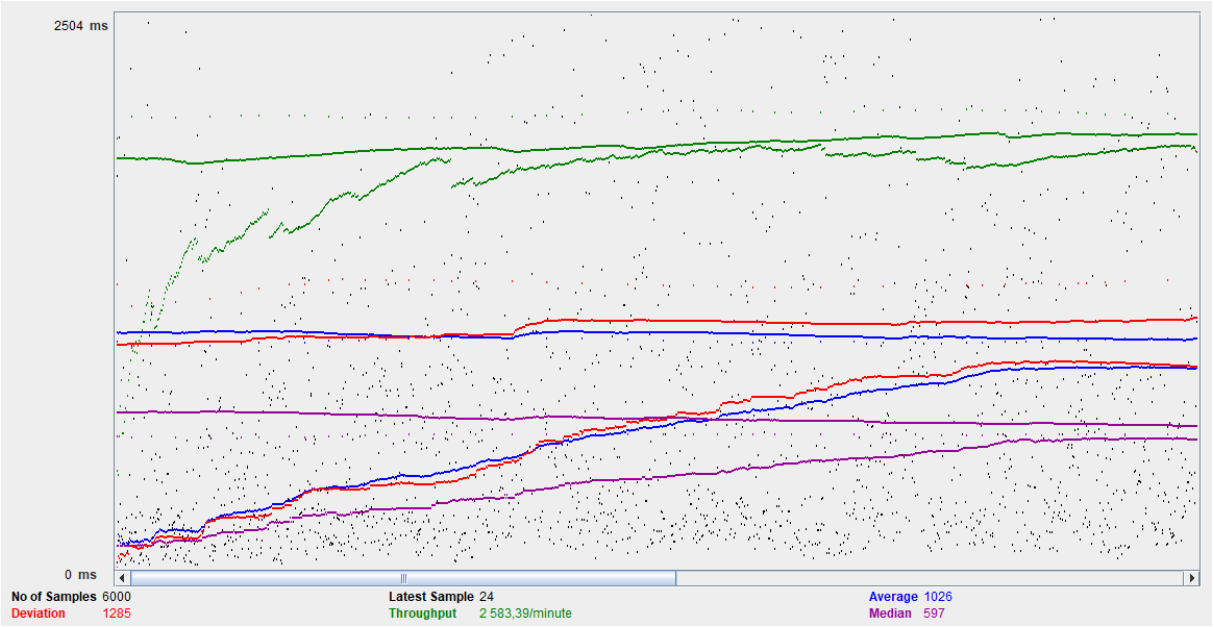
Výsledky zátěžových testů



Obrázek 7 - Výsledky testování výkonu při 10 uživatelích



Obrázek 8 - Výsledky testování výkonu při 20 uživatelích



Obrázek 9 - Výsledky testování výkonu při 50 uživatelích