FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Sinis - systém pro kolejní klub Sincoolka
Student:	Vojtěch Kváš
Vedoucí:	Ing. Filip Glazar
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství 2021
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Cílem této práce je důkladně zanalyzovat softwarové řešení kolejního klubu Sincoolka. Na základě této analýzy, která bude mimo jiné obsahovat i popis procesů potřebných pro provoz kolejního klubu, navrhněte nový podpůrný software pro tento kolejní klub. Realizujte pouze serverovou část aplikace. Pro budoucí implementaci klientské části aplikace připravte kompletní dokumentaci. Ideálně postupujte dle následujících kroků:

1) Analyzujte stávající SW řešení
2) Popište doménu a procesy, které v systému probíhají
3) Navrhněte vhodné SW řešení pro nový systém, který pokryje stávající funkcionalitu
4) Implementujte klíčové moduly serverové části aplikace včetně vhodně zvolených testů
5) Vytvořte dokumentaci pro následující implementaci klientské části řešení

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 23. ledna 2024 v Praze.

Bakalářská práce

SINIS – SYSTÉM PRO KOLEJNÍ KLUB SINCOOLKA

Vojtěch Kváš

Fakulta informačních technologií
Katedra teoretické informatiky
Vedoucí: Ing. Filip Glazar
16. května 2024

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Rád bych poděkoval vedoucímu práce Ing. Filipu Glazarovi za jeho čas a rady při vypracování této bakalářské práce. Dále bych rád poděkoval Bc. Pavlu Valachovi za pomoc s analýzou starého projektu a upřesnění požadavků systému. V neposlední řadě bych chtěl poděkovat rodině a přátelům za jejich podporu.

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.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 16. května 2024

Abstrakt

Tato bakalářská práce se zabývá analýzou, návrhem a implementací serverové části informačního systému pro kolejní klub Sincoolka. Klub nyní provozuje informační systém, avšak současné řešení má nedostatky v bezpečnosti, konkrétně SQL injekce a XSS. Nedokonalosti systému spočívají v členitosti a přehlednosti kódu. Po popisu stávajícího systému následuje analýza, návrh, implementace, testování a API dokumentace nového řešení. Serverová část systému je implementována pomocí frameworku Spring, který poskytuje REST API, jazyku Java a databázového systému PostgreSQL. Systém zpracovává osobní údaje o členech, ukládá a páruje platby se členem za účelem vytvoření členství nebo nabití konta pro tisk. Platby vznikly prostřednictvím platební brány nebo bankovním převodem. Systém umožňuje spočítat cenu vytisknutí dokumentu na kolejní tiskárně. Dále pomáhá s průběhem voleb do zastupitelstva koleje a klubu. V neposlední řadě podporuje správu místní sítě, především ukládá sítová zařízení a určuje, které se může připojit.

Klíčová slova informační systém, backend, kolej, Sincoolka, Spring, Java, PostgreSQL

Abstract

This bachelor thesis deals with the analysis, design, and implementation of the server-side component of the information system for the Sincoolka dormitory club. The club currently operates an information system, but the current solution has vulnerabilities such as SQL injection and XSS. Deficiencies of system are in code modularity and clarity. The thesis includes analysis, design, implementation, testing and API documentation of the new solution. The server-side component of the system is implemented using the Spring framework, which provides the REST API of the system, Java language and database system PostgreSQL. The system processes personal information about members, stores and matches payments with the member, in order to create a membership or charge an account for printing. Payments were made through a payment gateway or bank transfer. The system allows you to calculate the price of printing a document on a dormitory printer. It also helps with the organization of elections to the college council and the club. Last but not least, it supports local network management, mainly storing network devices and determining which ones can connect.

Keywords information system, backend, dormitory, Sincoolka, Spring, Java, PostgreSQL

Seznam zkratek

API	Application Programming Interface
DRY	Don't Repeat Yourself
DTO	Data Transfer Object
HTML	Hypertext Markup Language
JPA	Java Persistence API
PHP	Hypertext Preprocessor
SQL	Structured Query Language
XSS	Cross-site scripting
YAML	YAML Ain't Markup Language

Úvod

Někteří vysokoškolští studenti žijí na kolejích, především ti, kteří mají školu daleko. Na kolejích následně vznikají kolejní kluby, které pořádají společenské akce, poskytují internetové připojení, spravují posilovny a vytváří sociální komunity.

Poskytování nejrůznějších služeb vyžaduje určitou organizační náročnost, a proto se začaly vyvíjet informační systémy, které se správou klubů pomáhají. Především se jednalo o kluby, které měly nějaké zákonné povinnosti spjaté se správou financí a poskytováním internetu. Jednou z nutných podmínek pro vznik kolejního systému je mít členy, kteří ho jsou schopni a ochotni vytvořit a spravovat, nebo klub musí mít dostatečné finanční rezervy a systém svěřit externí firmě.

Podmínky pro vznik kolejního systému má i kolejní klub Sincoolka, který působí na Sinkuleho a Dejvické koleji. Na těchto kolejí jsou ubytováni mimo jiné studenti z fakulty informačních technologií, fakulty elektrotechnické i fakulty jaderné. Především tito studenti se stávají aktivními správci sítě a informačního systému. Společnými silami je systém na Sinkuleho koleji vyvíjen už 10 let, bohužel s časem nevyklenul k dokonalosti, ale nasbíral nedostatky.

Výsledek této bakalářské práce nahradí stávající systém pro správu chodu klubu Sincoolka. Z této práce budou mít prospěch především správci klubu Sincoolka, protože stávající systém je neudržitelný, jelikož nemá ani jeden test, kód obsahuje duplicity a není konzistentní. Tuto práci ocení i prostí členové klubu, jelikož nový systém jim přidá nové funkcionality, které dříve měli k dispozici pouze správci sítě, jako je například přidání uživatelského zařízení do sítě. Ostatní kolejní kluby se mohou inspirovat návrhem a funkcionalitami nového systému, které jsou v této bakalářské práci navrženy.

Téma bylo vybráno na základě ubytování autora této bakalářské práce na Sinkuleho koleji, který je platným členem klubu Sincoolka. Po zběžném prozkoumání stavu kolejního systému bylo rozhodnuto, že je třeba podniknout určité kroky. Přestože je kolejní klub Sincoolka nezisko-vou organizací, má mnoho společných prvků s komerční sférou. Prozkoumání této problematiky přinese zajímavý přínos do života.

První kapitola je věnovaná cílům této bakalářské práce. Další kapitola se zabývá analýzou současného softwarového řešení kolejního klubu Sincoolka. Následuje popis domény a funkční požadavky klubu Sincoolka. Čtvrtá kapitola se zabývá návrhem nového kolejního systému, na kterou navazuje pátá kapitola implementace aplikace. Po implementaci je na řadě dokumentace pro podporu vývoje klientské části aplikace. Projekt završí testy, které otestují funkčnost pomocí API.

Kapitola 1
Cíle

Cílem této bakalářské práce je analyzovat a popsat současný software pro provoz kolejního klubu Sincoolka. Inspirovat se starým řešením pro vytvoření analýzy nového řešení. Vhodné je se vyvarovat předešlým chybám a neopakovat nedostatky návrhu současného řešení.

Dalším cílem je na základě analýzy navrhnout nový systém, který minimálně pokryje současnou funkcionalitu. Už v návrhu je nutno počítat s případnými budoucími změnami a rozšířením služeb klubu Sincoolka.

Dílčím cílem je implementace serverové části nově navrženého systému. Implementace musí být především konzistentní a rozšiřitelná.

Následujícím cílem je otestování implementované serverové části, aby budoucí vývoj klientské části probíhal bez potíží.

Posledním cílem je vytvořit dokumentaci, která především pomůže s budoucí implementací klientské části. Bude se tedy jednat o API dokumentaci.

Kapitola 2

Analýza současného softwarového řešení

Tato kapitola je věnována analýze současného softwarového řešení. Soupis funkčních požadavků lze nalézt v následující kapitole, kde bude každý požadavek obsahovat atribut, zdali se vyskytuje i ve staré implementaci.

Kolejní systém je průběžně vyvíjen už 10 let. Do projektu přispívají neplacení dobrovolníci a nadšenci, kteří obvykle s opuštěním klubu Sincoolka opouští i tento projekt. Tyto skutečnosti se odráží v návrhu aplikace, kvalitě i přehlednosti kódu. Aplikace je napsána jako jeden monolit, je však snaha tento monolit rozdělit na část pro administrátory a část pro prosté uživatele.

Systém má na starosti ukládání osobních údajů členů klubu Sincoolka. Členovi zobrazuje jeho členské příspěvky. Ukládá a zaznamenává změny osobních internetových zařízení. Pomáhá s organizací voleb. Spravuje členské karty do posilovny a umožňuje členům si za poplatek vytisknout dokumenty na tiskárně.

2.1 Technologie současného softwarového řešení

Základní technologie se od počátku vývoje systému nezměnily, spíše postupně přibývaly. Lze najít v projektu soubory, které se od prvního nasazení neupravily. Především tyto soubory obsahují bezpečnostní nedostatky a jsou nepřehledné.

2.1.1 PHP

Hlavní část systému je napsána v PHP. Tímto jazykem je implementováno okolo 90 % funkčních požadavků.

PHP je zkratka pro hypertextový preprocesor. Jedná se o široce používaný open source skriptovací jazyk, který je vhodný pro vývoj webových stránek a může být vložen do HTML. Tento jazyk je odlišen od klientem prováděného JavaScriptu tím, že je vykonáván na serveru. Server převede PHP na HTML, které je odesláno klientovi. Klient obdrží výsledky běhu tohoto skriptu, ale neví, jak vypadá PHP kód. [1]

Hlavní přednost spočívá v tom, že je pro začátečníka extrémně jednoduchý na naučení, ale i přesto nabízí mnoho pokročilých funkcí pro profesionální programátory. Ačkoli se vývoj tohoto jazyka zaměřuje na serverové skriptování, jde s ním udělat mnohem více. [1]

PHP je mimořádně flexibilní nástroj pro serverové skriptování, které umožňuje sběr dat z formulářů, generování dynamického obsahu stránek nebo práci s cookies. Tato schopnost zahrnuje

tři hlavní oblasti: serverové skriptování, skriptování z příkazové řádky a psaní desktopových aplikací pomocí PHP-GTK. Dostupnost zajišťují všechny hlavní operační systémy a podpora je na většině dnešních webových serverů. Kromě toho, že umožňuje výstup HTML, PHP zvládne i bohaté typy souborů, šifrování dat nebo odesílání e-mailů. Jednou z jeho nejsilnějších vlastností je podpora široké škály databází a komunikace s jinými službami pomocí různých protokolů. Existuje také nabídka užitečných funkcí pro zpracování textu a mnoho dalších rozšíření, které mohou poskytnout ještě širší možnosti použití. [2]

2.1.2 Nette

V projektu lze nalézt framework Nette, využíván je minimálně, především se používá pro odesílání elektronické pošty.

Nette je vysoce výkonný PHP framework, který přináší výjimečnou kombinaci pohodlí, bezpečnosti a efektivity do procesu vývoje webových aplikací. Tento framework umožňuje vývojářům soustředit se na esenciální aspekty tvorby aplikací. Nette vede k psaní čistého kódu a efektivnímu využívání znovupoužitelných komponent. Nette rovněž zdůrazňuje bezpečnostní aspekty vývoje, což se projevuje v integrovaných bezpečnostních opatřeních. Hlavním mottem je: „méně kódu = dostatek bezpečí". S flexibilní sadou balíčků, které Nette nabízí, mají vývojáři možnost využít specializované nástroje, jako je například ladící nástroj Tracy či šablonovací systém Latte [3]. Tato vlastnost umožňuje přizpůsobit framework specifickým potřebám projektu a maximalizovat jeho efektivitu. Kromě toho, že Nette poskytuje stabilní a ověřené řešení, dokáže držet krok s nejnovějšími trendy ve vývoji webových aplikací a zůstává agilní díky pravidelným aktualizacím a podpoře nových verzí PHP. Jeho popularita mezi profesionály a významnými společnostmi je důkazem jeho vynikajícího výkonu a spolehlivosti. [4]

2.1.3 Latte

Latte je šablonovací systém. V tomto projektu je přibližně 25 % stránek vykreslováno pomocí Latte, zatímco zbylé stránky nevyužívají žádné šablony.

Latte [3] je šablonovací systém pro PHP, který efektivně brání proti útokům typu Cross-Site Scripting neboli XSS [5] a zajišťuje bezpečné zobrazování uživatelských dat na stránkách. Poskytuje jednoduchou syntaxi podobnou PHP s úsporným režimem pro zkrácený zápis. Latte usnadňuje tvorbu webových aplikací bez nutnosti se naučit nový jazyk. Plná podpora vývojových prostředí a efektivní ladění dělají z Latte spolehlivého partnera pro tvorbu rychlých, bezpečných a intuitivních webových stránek.

2.1.4 PostreSql

Pro ukládání dat je zvolen databázový systém PostreSql. Veškerá data týkající se aplikace Sinis jsou uložena v jedné databázi.

PostgreSQL [6] je výkonný open source objektově-relační databázový systém, který používá a rozšiřuje jazyk SQL a nabízí mnoho funkcí pro bezpečné ukládání a škálování datových zátěží. Jeho původ sahá až do roku 1986 jako součást projektu POSTGRES na univerzitě v Berkeley. Za svou více než 35letou aktivní existenci si PostgreSQL vydobyl silnou reputaci díky své spolehlivosti, rozsáhlým funkcím a oddané komunitě.

2.1.5 JavaScript

Dále se v projektu objevuje JavaScript [7]. Jedná se o lehký interpretovaný programovací jazyk s funkcemi první třídy [8]. Je nejznámější jako skriptovací jazyk pro webové stránky. Mnoho ne-webových prostředí ho také využívá, jako je například Node.js, Apache CouchDB a Adobe

Acrobat. JavaScript je prototypově orientovaný dynamický jazyk, podporující objektově orientované, imperativní a deklarativní styly. Dynamické schopnosti JavaScriptu zahrnují konstrukci objektů za běhu. Standardy pro JavaScript jsou Specifikace jazyka ECMAScript ECMA-262 a Specifikace mezinárodního API pro ECMAScript ECMA-402. Java [9] a JavaScript jsou dva velmi rozdílné jazyky, i přestože jsou oba objektově orientované. Hlavní rozdíl spočívá v typování. JavaScript je dynamicky typovaný a Java je staticky typovaná. Další podstatný rozdíl spočívá v kompilaci. Java patří mezi kompilované jazyky, ale JavaScript je interpretovaný skriptovací jazyk.

2.1.6 jQuery

Čistý Javascript je zde v menšině, především se využívá jQuery [10] knihovna. Tato knihovna usnadňuje práci s JavaScriptem a manipulaci s HTML a CSS na webových stránkách. Jedná se o open-source knihovnu, která poskytuje jednoduchý a konzistentní způsob, jak pracovat s DOM, což je zkratka pro Document Object Model. Dále podporuje události a asynchronní načítání dat.

Hlavním cílem jQuery je zjednodušit vývoj interaktivních a dynamických webových stránek tím, že poskytuje jednotný a křížově kompatibilní způsob manipulace s DOM, který je nezávislý na konkrétním webovém prohlížeči. To znamená, že vývojáři nemusí psát oddělený kód pro každý prohlížeč, což značně usnadňuje a zrychluje vývoj webových aplikací. [10]

2.1.7 DataTables

DataTables je někdy označován jako jQuery DataTables, jedná se o populární jQuery plugin, který umožňuje snadno manipulovat s tabulkami dat na webových stránkách. Tento plugin poskytuje bohaté možnosti pro filtrování, řazení a stránkování tabulkových dat přímo v prohlížeči. [11]

Hlavní výhodou použití DataTables je jeho jednoduchá integrace do existujícího webového kódu. Stačí přidat odkaz na knihovnu DataTables, definovat HTML tabulku s daty a inicializovat DataTables plugin pomocí jednoduchého volání jQuery funkce. [11]

DataTable plugin umožňuje dynamicky načítat data z různých zdrojů, jako jsou JSON soubory, XML dokumenty nebo nativní HTML tabulky. To umožňuje flexibilní manipulaci s daty a snadnou integraci s různými technologiemi na straně serveru. [11]

Další klíčovou funkcí je možnost filtrování dat pomocí různých kritérií, jako jsou textová pole, rozbalovací seznamy nebo časová rozmezí. Uživatelé mohou rychle vyhledávat a filtrovat data bez nutnosti načítání stránky znovu. [11]

2.1.8 Git

Pro verzování se používá git [12]. Jedná se o distribuovaný systém správy verzí, který se používá ke sledování změn v kódu a spolupráci mezi vývojáři. Byl vyvinut Linusem Torvaldsem v roce 2005 a od té doby se stal jedním z nejpoužívanějších verzovacích systémů na světě.

Git umožňuje vývojářům ukládat svůj kód do repositářů, sledovat změny, vytvářet větve pro experimentování. Umožňuje izolaci nových funkcionalit, které lze poté sloučit zpět do hlavní vývojové větve. Tímto způsobem poskytuje kontrolu nad verzemi kódu a umožňuje spolupráci mezi vývojáři. [12]

2.2 Databázová vrstva

Databázová vrstva představuje klíčovou součást aplikace, která se stará o ukládání a správu dat. Jejím hlavním cílem je zajistit efektivní manipulaci s daty a udržovat jejich konzistenci a integritu.

2.2.1 Databázový systém

Pro ukládání dat je použit relační databázový systém, konkrétně PostgreSQL [6]. PostgreSQL byl zvolen pro svou robustnost a podporu rozsáhlých datových operací.

2.2.2 Databázové schéma

Databázové schéma [13] je struktura, která definuje organizaci dat v databázi. Zahrnuje tabulky, které jsou základními jednotkami dat, a sloupce, které určují vlastnosti dat v každé tabulce. Dále schéma definuje vztahy mezi tabulkami. Pro zajištění integritních omezení se používají primární a cizí klíče, které identifikují záznamy a určují vztahy mezi nimi. Kromě toho mohou být ve schématu definována omezení integrity dat, jako jsou omezení unikátnosti nebo omezení referenčního klíče, která pomáhají udržovat konzistenci a správnost dat v databázi. Databázové schéma je základem pro návrh a správu databází.

Strukturu databáze současného řešení lze nalézt ve složce sql i s jejími migracemi. Základní schéma je navrženo správně, ale s přibývajícími funkcionalitami postupně degradovalo. Databázové schéma je zobrazeno pomocí obrázku A.1.

Normální formy

Normální formy [14] jsou respektovány, až na menší výjimky. První normální forma je zde bezpodmínečně dodržována, protože každý sloupec v tabulce obsahuje pouze atomické hodnoty. Druhá normální forma je vždy splněna, protože se používají generované primární klíče. Třetí normální formu porušuje tabulka c_university, kde jsou uloženy názvy univerzit a fakult. Fakulta nezávisí přímo na klíči, ale jedná se o samostatnou entitu. Stejné tvrzení se může říci o tabulce socket, která obsahuje switch a port, ale v aplikační vrstvě se tabulka socket nevyžívá a může být tedy smazána.

Volby a hlasy

Hlavním nedostatkem databázového návrhu je, že neukládá hlasy ke konkrétním volbám, ale pouze ke členovi, což umožňuje ověřit, zdali uživatel volil či nikoliv, ale nelze si zobrazit hlasy z předchozích voleb, protože před začátkem nových voleb se musí tabulka voted vyprázdnit, aby se nové hlasy mohly vkládat. Nelze tedy dohledat, jaké volby proběhly, kdo v nich volil a kdo nikoliv. Případné volby stejného typu, které by měli časový konflikt nejde realizovat, protože by došlo v tabulce voted ke kolizím.

Role a odpojení

Dalším nedostatkem tohoto návrhu je ukládání administrátorských práv, která jsou vázána na členství. Pokud se členství prodlužuje, neboli člen si zakoupí nebo obdrží nové členství na další semestr je nutné práva překopírovat do nového členství.

Také jsou v tabulce membership sloupce dc_gym a dc_net, jejich funkcionalitou je zamezení přístupu do posilovny a odstřihnutí člena od internetu. Pro tyto sloupce platí stejná pravidla jako pro administrátorská práva, s tím rozdílem, že jejich případné špatné zkopírování člen nenahlásí a s radostí si bude užívat přístupu do posilovny nebo k internetu. Jedná se rozhodně o špatný návrh, protože zákazy přístupu jsou dočasné, například na 10 dní. Nelze také zpětně dohledat kdo, kdy, co a na jak dlouho byl zablokován.

Logování změn

Kolejní klub poskytuje internet, a proto má povinnost zpětného vyhledání uživatele v případě provádění kriminální činnosti. Z toho důvodu se logují změny internetových zařízení v tabulce

nic_log. Loguje se pomocí triggerů. Logovací tabulku by šlo odstranit v případě zakázání provádění změn. Avšak systém neumožňuje členovi přidávat zařízení, ale pouze upravovat. Přidávat zařízení může pouze správce sítě, kterému by díky smazání logovací tabulky přibyla práce. Nelogují se pouze změny internetového zařízení, ale také se zaznamenávají úpravy přístupových karet do posilovny, aby se případně dal dohledat člen, který si odnesl vybavení posilovny.

Tabulka variable

V databázi se nachází tabulka variable se sloupci value a variable. Využití spočívá v ukládání konfiguračních atributů. Pro ukládání konfiguračních proměnných by bylo vhodné využít konfigurační soubor. Pokud proměnná nepatří do konfiguračního souboru, tak lze vytvořil samostatnou entitu, která bude jednoznačně pojmenována, nebo problém vyřešit přidáním atributu do existující tabulky.

Duplicita hesel

V tabulce member se nachází sloupce pass a pass2, oba representují heslo, což je redundantní. Avšak obě hesla se používají v následující aplikační vrstvě, jedno heslo je zašifrované pomocí MD4 a druhé za pomoci SHA-512. MD4 hash se používá pro připojení k WIFI a hash SHA-512 byla přidána pro autorizaci do systému, bohužel nikdy nebyla plně implementována.

Další nedostatky

Ve schématu se nalézají tabulky, které se nepoužívají, jako je například old_member nebo ad_user_status.

Nelze přehlédnout absenci cizího klíče v tabulce kelimky, konkrétně sloupec id_member má zajistit spojení s tabulkou member.

Pojmenování tabulky kelimky bije do očí, poněvadž se jedná o jediné české slovo v celém schématu.

Dále sloupec id_ref_member v tabulce member by měl být cizím klíčem a vázat se na sloupec id_member. V současném systému sloupec id_ref_member nemá žádné využití. Původně se zamýšlela funkcionalita, která by umožnila přidat uživatele, jenž není členem klubu Sincoolka. Daný uživatel by chodil se členem klubu do posilovny, pravděpodobně by se jednalo o kamaráda. Sloupec id_ref_member měl zajistit vazbu mezi členem a jeho kamarádem.

2.3 Aplikační vrstva

Aplikační vrstva, někdy nazývaná také jako Business Layer nebo Logická vrstva, představuje klíčovou část architektury softwarové aplikace. Tato vrstva obsahuje implementaci byznysových pravidel, logiky a procesů, které definují chování aplikace a zajišťují realizaci funkcí potřebných k dosažení cílů systému. Hlavním účelem aplikační vrstvy je poskytnout rozhraní mezi uživatelským prostředím a datovou vrstvou. To znamená, že aplikace přijímá požadavky od uživatelů, provádí potřebné operace pomocí logiky aplikační vrstvy a poté přistupuje k datům uloženým v databázové vrstvě.

V aplikační vrstvě se nachází logika pro zpracování dat a vykonávání operací souvisejících s byznysem. To může zahrnovat validaci uživatelského vstupu, zpracování podnikových pravidel, provádění výpočtů, transformaci dat a další manipulace potřebné pro splnění požadavků aplikace. Důležitou vlastností této vrstvy je oddělení byznysových pravidel od ostatních částí aplikace, což usnadňuje údržbu, rozšiřitelnost a znovupoužitelnost kódu.

Aplikační vrstva bývá obvykle umístěna ve složce src, ale v tomto projektu je umístěna v kořenovém adresáři. Ideální by bylo vytvořit adresář src, ve kterém by byly umístěné složky s kódem, nyní se složky s kódem míchají se složkami sql a docker.

Aplikační vrstva systému Sinis je logicky dělena na část určenou pro prostého člena klubu. Zde si člen může prohlížet a měnit svoje osobní údaje, například heslo. Dále má možnost vytvořit svůj hlas ve volbách, upravit svoje zařízení určené pro připojení k internetu. Vytvoření nového internetového zařízení systém bohužel nepodporuje, nové zařízení smí vytvářet pouze správce sítě. V neposlední řadě, si zde uživatel může prohlédnout svoji historii finančních příspěvků, tisku a stavu konta, na kterém jsou uloženy kredity pro vytisknutí dokumentů na tiskárně.

Další část se zabývá administrací. Do administrace spadá zaregistrování nového člena klubu, přidání nového zařízení pro připojení k internetu, vytvoření nové karty do posilovny a procházení záznamů přístupů do posilovny. Správa voleb zahrnuje příjem kandidátek a označování uživatelů, kteří hlasovali pomocí urny. Dále je nutné spojit nespárované platby a členy. V neposlední řadě se přidělují kelímky členům klubu Sincoolka.

V menší složce je uložen kód pomáhající s před-registrací nového uživatele. K systému patří nejrůznější skripty, které vykonávají různorodé úkoly, jako je například stahování plateb z API banky.

2.3.1 Modul pro členy

Ve složce usr_iface se nachází aplikační logika pro správu osobních údajů prostého člena. Modul zobrazuje členovi jeho údaje a informace zanesené v systému. Některé informace může editovat, jako například síťové zařízení nebo jeho fakultu. Za pomoci systému se dá hlasovat ve volbách.

Na složce usr_iface lze ocenit strukturu, jeden soubor representuje jednu webovou stránku nebo funkcionalitu. Lze kritizovat míchání logiky aplikace a dotazů do databáze. Ve složce usr_iface se data připraví pro latte šablony. Šablony jsou umístěné ve složce template, v tomto případě ve složce template/usr_iface. S pomocí šablony se data vykreslí uživateli. V šabloně Latte nejsou žádné výrazné nedostatky.

Za nedostatek lze považovat implementaci vytvoření hlasu v souboru usr_iface/volby.php, jehož úryvek si lze prohlédnout ve výpisu kódu 2.1. V tomto souboru se složitými podmínkami rozděluje přidání hlasu do tabulky voted, aby se pak následně do tabulky vložily příznaky pravda a nepravda pro typy voleb. Například v podmínce se podle proměnné $sin rozdělí tok na dvě možnosti. Pokud je hodnota $sin pravda, tak se vloží do sloupce sin v databázi hodnota pravda a v nepravdivé větvi se vloží nepravda. Bylo by vhodné větvení smazat a $sin použít jako vstupní proměnou do databázového příkazu pro vytvoření hlasu.

```php
<?php
foreach ($_POST as $key => $value) {
    if ($key != "send") {
            $result = $SQLnew->fetch("UPDATE candidates SET
            votes=votes+1 WHERE id_member = ?", $key);
    }
}
if ($sin) {
    if ($board) {
        $query = "INSERT INTO voted (id_member, board, sin, time_stamp)
        VALUES (?, 't', 't', date_trunc('seconds',now()))";
        $res = $SQLnew->query($query, $id_member);
    }
    else {
        $query = "INSERT INTO voted (id_member, board, sin, time_stamp)
        VALUES (?, 'f', 't', date_trunc('seconds',now()))";
        $res = $SQLnew->query($query, $id_member);
    }
}
?>
```

Výpis kódu 2.1 Ukázka kódu ze souboru usr_iface/volby.php

2.3.2 Modul pro administrátory

Administrátoři mohou prohlížet data patřící členům, případně mohou členovi přidat síťové zařízení, kartu pro přístup do posilovny a mnoho dalšího.

Pro zajímavost lze v projektu nalézt dva indexy. První je umístěn v kořenovém adresáři, druhý je ve složce ar. Zkratka ar znamená admin registration. Zde jsou implementovány interakce správců se systémem. Složka ar si žije vlastním životem. Nalézá se zde css.css pro stylování webové stránky, ikona favicon.ico, samozřejmě stejná ikona se stejným názvem je uložena i v kořenovém adresáři.

Dále jsou zde soubory týkající se menu, které by bylo vhodné přemístit do stejnojmenné složky. Ideálně do nově vytvořené složky /templates/ar/menu, protože všechny až na jeden soubor obsahují čisté HTML. Soubor ar/menu_rel.php kromě HTML navíc vlastní PHP a JavaScript.

Ostatní PHP soubory se nalézají v adresářích ar/adm_login, ar/pages a ar/scripts. V jednom souboru není problémem najít HTML, PHP, JavaScript a dotaz do databáze, příkladem souboru, který posbíral všechny vrstvy, je /ar/pages/mm_search.php.

I správci se některé stránky vyrendrují pomocí latte, bohužel se jedná o ojedinělé případy. Kód starající se o administraci působí nepřehledným dojmem. Vše napsáno špagetovým způsobem, lze to především pocítit v souborech, které mají okolo 400 řádků. Právě proto je na Sinkoleho koleji pouze jeden odborník, který se v systému Sinis plně vyzná.

V adresáři ar/pages/statistiky se nachází soubory, které správcům ukazují nejrůznější grafy. Pro vykreslení grafů se používá google.visualization.DataTable(). Například soubor ar/pages/statistiky/nation_members.php zobrazuje národnost a počet členů.

2.3.3 Modul pro přihlášení člena

Tento modul se nachází ve složce usr_login. Tato složka má za úkol se postarat o přihlášení a obnovu hesla uživatele. Kód v této složce se buďto vůbec nevyužívá, a ten, co se využívá je někdy duplicitní s přihlašovacím systémem ve složce ar. Ideální by bylo mít kód týkající se přihlašování v jedné složce a předcházet tak duplicitám kódu.

2.3.4 Modul pro před-registraci člena

S cílem urychlit proces registrace byla zavedena možnost, která umožňuje žadatelům o členství vyplnit před-registrační formulář, díky němuž si registrátor lehce načte data zájemce o členství. Tento modul se nachází ve složce usr_registrace. Celý proces registrace nového uživatele není jednoduchý a špagetově napsaný kód rozhodně neulehčí případné modifikace tohoto procesu.

V souboru usr_registrace/registrace_odeslani.php si kritiku zaslouží provedení transakce, která začne několika příkazy SELECT a následně provede příkaz INSERT INTO do jedné jediné tabulky member, následuje příkaz COMMIT. Na dalších řádcích se nastavuje a ukládá heslo uživatele, přidává zařízení pro připojení k internetu a vytváří členství na 5 dní, aby uživatel měl přístup k internetu, a vše ukončuje odeslání emailu. Tuto nedokonalost zobrazuje výpis kódu A.1.

Novému členovi se tedy může vytvořit účet, ale nebude mít funkční heslo ani možnost připojení k internetu a neobdrží email o registraci, ale když se pokusí o vytvoření nového účtu, bude mu systém hlásit, že jeho email je už zabraný.

2.3.5 Modul pro validaci emailu

Modul pro validaci emailu obsahuje jeden soubor validmail/validmail.php, jehož funkcionalitou je ověřit, zdali je email platný. Této funkcionality využíval informační systém Silicon Hill, který nabízel různé kurzy i pro členy klubu Sincoolka, proto Silicon Hill potřeboval ověřit platnost emailové adresy. Tato funkcionalita se již dnes nevyužívá a tento adresář by bylo vhodné smazat.

2.3.6 DataTables a jQuery_tables

DataTables umožňují efektivně prohledávat tabulku, jak již bylo dříve zmíněno v technologiích. Prohledávání a řazení najde využití především v administrátorské části aplikace, kde pomáhá zobrazovat a prohledávat data.

Tento projekt má velký problém s duplicitou, důkazem jest DataTables, které jsou v projektu uložené dvakrát, jednou pod jménem DataTables, podruhé pod jménem jQuery_tables.

2.3.7 Komunikace se čtečkou

Systém pro komunikaci se čtečkou karet je umístěn ve složce card_system. Systém karet umožňuje přečíst přístupy do posilovny a kontrolovat čtečku, jestli je aktivní.

2.3.8 Nástroje

V kořenovém repositáři je uložena složka utils, ve které se nalézají pomocné funkce a užitečné nástroje. Ve složce jsou uloženy konfigurační soubory pro připojení k databázi a nastavení odladovače též nazývaného debugger. V souboru utils/auth.php jsou definované funkce pro přihlášení, odhlášení, zjištění role uživatele a mnoho dalšího. Tyto funkce podporují DRY princip. Zkratka DRY znamená „Don't repeat yourself", neboli „Neopakuj se". Aby DRY princip platil, se musí funkce používat, k čemuž nedošlo například v souboru usr_login/zapom_heslo.php, kde by se

krásně dala využít funkce update_password, která jako parametry přijímá id člena klubu a nové heslo. Úplně to samé by se dalo říci o kódu v souboru ar/pages/usr_pass_reset.php.

Soubor nazvaný utils/bitwise.php se naštěstí už nepoužívá a měl by být smazán. V minulosti se jednalo o implementaci přiřazení rolí členovi klubu pomocí bitové masky. Od tohoto řešení se upustilo, protože implementace byla nepřehledná a složitá.

S přepínáním jazyku stránky vypomáhá utils/dict.php, menší ukázku lze shlédnout ve výpisu kódu 2.2. Slova a věty jsou uloženy pomocí asociativního pole. Jak už to v tomto systému bývá, klíče nejsou konzistentně pojmenovávány, jeden je anglicky a bez problému následuje český název. Největším nedostatkem je vkládání struktury dokumentu do hodnot proměnných.

Soubor utils/utils.php ukládá nejrůznější funkce, které jsou většinou používány pro validaci. Funkce verifyCARD vždy vrátí true a má komentář „// TODO STUB". Největší problém tkví v tom, že je používána ve dvou souborech /ar/pages/card_member_card_script.php a /ar/-pages/usr_member_card_script.php.

```
<?php
'volby_check_your_room_1' => '<strong>Důležité:</strong> Podle našich záznamů
bydlíte v pokoji ',
'volby_check_your_room_2' => '.<br>Není-li tomu tak, prosím,
<a href="https://sinis.sin.cvut.cz">přihlaste se do SINISu</a>
a změňte si pokoj ještě před hlasováním, v sekci Uživatelské údaje. Děkujeme!',

'volby_check_your_room_not_logged_in' => '<strong>Důležité:</strong> Prosím,
<a href="https://sinis.sin.cvut.cz">přihlaste se do SINISu</a>
a zkontrolujte si pokoj v sekci Uživatelské údaje. Děkujeme!',

// footer
'footer_gdpr' => '<a href="https://su.cvut.cz/gdpr">Zásady ochrany
osobních údajů</a>',
'footer_sin' => '<a href="https://www.sincoolka.cz">Stránky klubu Sincoolka</a>',
'footer_regulations' => '<a href="https://wiki.sin.cvut.cz/verejne/predpisy">
Stanovy a předpisy</a>',
?>
```

Výpis kódu 2.2 Ukázka kódu ze souboru utils/dict.php

2.3.9 Nekonzistentní pojmenovávání

Při prozkoumávání struktury projektu bylo identifikováno několik nedostatků, které by mohly přispět ke zmatení a neefektivnosti při správě kódu. Jednou z hlavních oblastí, která vyžaduje pozornost, je nedostatečná organizace souborů a složek v adresářové struktuře.

Názvy souborů a složek jsou pojmenovány anglicky za použití konvence snake case [15]. Tato konvence se však stoprocentně nedodržuje, příkladem jsou názvy složek jQuery_tables, validmail, DataTables. Český název souboru se občas objeví, třeba ve formě usr_iface/volby.php. Názvy proměnných se potýkají se stejnými problémy jako názvy souborů. Například v souboru usr_iface/volby.php se nachází proměnné $pocetListku a $candidates_board. Komentáře jsou z poloviny v češtině a z poloviny v angličtině. V současném stavu kódu by se pouze anglicky mluvící programátor stěží vyznal. Konzistence není silnou stránkou projektu. Celý kód je psán špagetovým způsobem, což rozhodně na přehlednosti nepřidá.

2.3.10 Transakce

V projektu chybí transakce, důkazem toho může být soubor /usr_iface/volby.php, ve kterém se inkrementuje kandidátův počet hlasů a poté je voliči vytvořen platný hlas. V případě chyby mezi navýšením počtu hlasů a přidělení hlasu, bude člen mít možnost hlasovat dvakrát a nelze tomu zabránit či zpětně dohledat o jakého člena se jednalo. Absence transakce je vyobrazena pomocí výpisu kódu 2.1.

2.3.11 Bezpečnostní nedostatky

SQLInjection

SQL injection [16] je zranitelnost webových aplikací, která umožňuje útočníkovi vložit škodlivý kód do dotazu určeného pro databázi. SQL je zkratka pro Structured Query Language, tento jazyk slouží pro správu databáze. S jeho pomocí se dotazuje a manipuluje s daty v databázi. Pokud webová stránka nedostatečně ošetří uživatelský vstup, může útočník vložit SQL kód, který se pak provádí na straně serveru spolu s ostatním SQL kódem.

SQL injection útoky mají různé účely, včetně získání neoprávněného přístupu k databázi, mazání dat, editace databáze nebo získání citlivých informací, jako jsou hesla uživatelů. Aby se zabránilo SQL injection útokům, je důležité, aby vývojáři webových aplikací prováděli dostatečné ošetření uživatelského vstupu, výsledkem by mělo být znemožnění vložení škodlivého kódu. [16]

Učebnicový příklad takovéto zranitelnosti lze nalézt v souboru ar/pages/usr_pass_reset.php, úryvek tohoto souboru se nachází ve výpisu kódu 2.3. Další zranitelnosti se nachází v souborech ar/pages/pay_unknow.php, ar/pages/utils_statusy.php a to zdaleka nejsou všechny, v posledně jmenovaném programátor doufá, že mu tam nikdo nepošle SQL injection. Data mu přichází z jiného souboru, obsah proměnných si zajišťuje a validuje pomocí komentáře, což se rovná nulové validaci.

```php
<?php
$nove_heslo = genRandomString(10);
$salt = '$6$' . bin2hex(random_bytes(6));
$hash = crypt($nove_heslo, $salt);
$ntlmHash = NTLMHash($nove_heslo);
$query = "UPDATE member SET pass = '$hash', pass2 = '$ntlmHash'
WHERE id_member='{$_GET["id"]}'";
$result = pg_query($query);

$result = pg_query("SELECT * FROM member WHERE id_member='" . $_GET["id"] .
"'");
$out = pg_fetch_array($result);

$zprava = "Admin reset your password<br>Your new password for SINIS: " .
$nove_heslo;
?>
```

Výpis kódu 2.3 Ukázka kódu ze souboru ar/pages/usr_pass_reset.php

Slabé šifrování hesel

Systém ochraňuje uživatelská hesla hashovacími funkcemi, bohužel nedostatečně. Jedno heslo uživatele je uloženo dvakrát, jednou pomocí MD4 a podruhé se využije SHA-512. SHA-512 hash je novější a kryptograficky silnější oproti MD4, i přestože je v systému solena, neposkytuje dostatečnou bezpečnost vůči brute-force útoku. Především nebyla ani k ukládání hesel určena, vhodnější je například bCrypt. Případný útočník si samozřejmě vybere MD4, vzhledem k tomu že není solena, může použít i slovníkový útok. MD4 hash je nutné ukládat, protože se využívá pro připojení k síti WI-FI. [17]

Cross-site scripting

Dalším bezpečnostním nedostatkem je Cross-site scripting, zkráceně XSS [5], se nalézá především v administrátorské části aplikace, která nevyužívá Latte šablony. XSS útoky vkládají škodlivé skripty do důvěryhodných webových stránek a zneužívají nedostatky ve validaci uživatelského vstupu. Útočníci tak mohou poslat škodlivé skripty nevinným uživatelům. Lze tímto způsobem získat přístup k citlivým informacím, jako jsou cookies nebo session tokeny. Tyto útoky mohou manipulovat s obsahem HTML a vést k různým následkům, od kompromitace účtů po přesměrování uživatelů či instalaci škodlivého softwaru.

Existují různé typy XSS útoků, včetně odražených též zvaných reflektovaných a uložených XSS. Reflektované útoky jsou ty, kde je vložený skript odražen z webového serveru, například v chybové zprávě, výsledku vyhledávání nebo jakékoliv jiné odpovědi, která obsahuje části nebo celý vstup odeslaný serveru jako součást požadavku. Reflektované útoky jsou doručeny obětem například v e-mailové zprávě, kliknutím na zákeřný odkaz, odesláním speciálně vytvořeného formuláře nebo dokonce jen procházením zákeřného webu. Vložený kód putuje na zranitelný web, který odrazí útok zpět do uživatelova prohlížeče. Prohlížeč poté provede kód, protože pochází z důvěryhodného serveru. [5]

Uložené útoky jsou ty, kde je vložený skript trvale uložen na cílových serverech, obvykle v databázi. Skript byl vložen útočníkem a následně se zobrazuje například v diskuzním fóru, v záznamech návštěvníků a v komentáři. Oběť poté nevědomě získá zlomyslný skript ze serveru. Uložený XSS je někdy také označován jako trvalý nebo Typ-II XSS. [5]

Kapitola 3
Analýza nového řešení

Analýza je jednou z prvních fází vývoje softwaru. V této kapitole je popsána analýza nového řešení informačního systému pro klub Sincoolka. Kapitola obsahuje krátkou analýzu stávajících řešení ostatních klubů, diagram aktivit, případy užití, funkční a nefunkční požadavky.

3.1 Analýza existujících řešení

Ještě před analýzou by mělo zaznít, proč se nepoužilo nějaké jiné už dříve implementované řešení. Jedním z argumentů by mohlo být, že vývoj a údržba informačního systému je nedílnou součástí klubu Sincoolka. Hlavním důvodem je především fakt, že na kolejní systém jsou kladeny unikátní požadavky. V současné době neexistuje projekt, který by dokázal splnit tyto specifické požadavky.

Informační systémy s podobnými funkčními požadavky se provozují na ostatních kolejních klubech ČVUT. Následující odstavce krátce popisují aktuální stav kolejních systémů a jejich vhodnost pro kolejní klub Sincoolka. Stav některých systémů stagnuje, jejich kondice je v roce 2018 popsána v 6. kapitole diplomové práce zabývající se informačním systémem Hlávkovi koleje [18].

3.1.1 Silicon Hill

Klub Silicon Hill působí na koleji Strahov a stále si udržuje a vylepšuje vlastní systém. Silicon Hill je přibližně 10 krát větší než Sincoolka, důsledkem je, že jejich systém robustnější a obsahuje funkcionality, které nenajdou využití na kolejním klubu Sincoolka, jako je například správa rozpočtů. Tento systém je nejžhavějším kandidátem pro klub Sincoolka. Toto řešení je odmítnuto ze stejných důvodů, jako bylo odmítnuté v práci zvané Registr členů klubu Hlávkova kole [18]. Především se jedná o vysoké nároky pro nasazení a údržbu. I přestože je systém robustní, bylo by nutné dodělat další funkcionality, jako například správa kelímků.

3.1.2 Pod-o-lee

Na kolejním klubu Pod-o-lee, sídlícím na koleji Podolí, se podařilo nasadit systém Hydra, který je napsán pomocí mikro-služeb. Kolejní klub Pod-o-lee je přibližně dvakrát větší než Sincoolka, i přesto však existuje domněnka, že mikro-služby nejsou správná cesta. Potíže mohou nastat při nasazení a při komunikaci mezi službami, monolit nepostihují tyto problémy. Na vývoji kolejního klubu se dle mých zkušeností podílí menší tým čítající okolo 5 lidí, není tedy třeba izolovat vývojové týmy.

3.1.3 Masařka

Masarykova kolej disponuje kolejním klubem Masařka. Dle zjištěných informací opustili informační systém DIS a implementovali nový systém.

Spring [19] framework byl použit pro implementaci serverové části aplikace. PostgreSQL [6] našel uplatnění pro ukládání dat. Vue.js [20] implementuje klientskou část aplikace. Klientská část se dotazuje na serverovou část pomocí rest API.

Celý systém je oproti požadavkům klubu Sincolka jednoduchý. Členové klubu Masařka platí pouze jeden příspěvek, aktuálně 1 000 Kč. Za tento příspěvek člen získá veškeré výhody klubu, včetně posilovny. V kolejních klubech bývá standardem si za posilovny připlatit. Systém umožňuje pouze platbu převodem. Celý platební systém by se musel v novém systému přepsat. Vstup do posilovny je řešen pomocí rezervace a fyzického vypůjčení klíče. V případě osvojení tohoto systému kolejním klubem Sincoolka, by ho bylo nutné skoro celý přepsat a dopsat nové funkcionality, což je mnohem více časově náročné než začínat od nuly.

Pro zajímavost je přiloženo schéma databáze A.2, kde je nesprávně rozdělen člen do dvou tabulek v závislosti na tom, zdali je registrován či nikoliv. Tento problém se dá vyřešit přidáním atributu. Konstantní a variabilní symbol jsou vždy čísla, proto by měli být reprezentovány pomocí datového typu, který reprezentuje číslo.

3.1.4 Buben

Kolejní klub Buben z Bubenečské koleje se rozhodl opustit informační systém KrlStin, za pomoci administrátorů ze Silicon Hill byl nasazen informační systém kolejního klubu Silicon Hill. Nedostatek programátorů neumožnil vytvoření ani správu vlastního systému.

3.1.5 BION

Fakultní klub BION, který se nachází na koleji v Kladně, stále využívá starý systém KrlStin a nadále se potýká s nedostatkem programátorů.

3.1.6 Registr členů klubu Hlávkovy koleje

Na Orlíku a na Hlávkově koleji úspěšně nasadili systém popsaný v diplomové práci Registr členů klubu Hlávkova kolej [18], tento systém se blíží potřebám a funkčním požadavkům kolejního klubu Sincoolka, proto mu bude věnována větší pozornost.

Systém pro správu Sinkuleho koleje je mnohem obsáhlejší díky správě kelímků a tisku. Tyto nové funkcionality by se musely dodělat, ale to i v případě vybrání těch nejrozsáhlejších klubových informačních systémů, jako například systém ze Sillicon Hill. V následujících odstavcích je popsána nevhodnost Registru členů klubu Hlávkovi koleje pro nasazení a vývoj. Hlavní rozdíly plynou z odlišného pohledu na problematiku provozu kolejního klubu.

Nesoulad lze najít ve zpracování plateb. Na obou kolejích se platí členství na půl roku, neboli na semestr. Členství se váže na semestr, a proto by měla v systému existovat entita semestr, na ni se budou platby krásně mapovat, protože každý semestr má atributy od kdy, do kdy přijímá platby. Další výhodou Sinis řešení je, že se dá zpětně jednoduše dohledat, kdo měl a neměl platné členství. Navíc kolejní klub Sincoolka přijímá platby kartou i převodem. Celá logika související s platbami by se musela přepsat.

Nedostatek tohoto systému je skryt v rolích. Dle schématu databáze každý uživatel má pouze jednu roli, což je pro Hlávkovu kolej dostačující. V Sincooleho klubu jsou role více členité a uživatel může mít několik rolí.

Další rozdíl se nachází v přístupu do posilovny. Přístup by měl být spojován s kartou, která má svého majitele. Přístup by se neměl mapovat přímo na člena, protože případné změny budou

problematické, například uživatel má několik různých karet nebo karta je vázána na roli a nikoliv na člena. Další nevýhodou je případné špatné mapování, nelze jednoduše zpětně opravit, pokud je karta smazána. Podobný argument se dá použít pro sítové sessions.

Přepsat informační systém pro Hlávkovu kolej a následně mu přidat funkcionality, je více časově náročné než začínat od začátku.

3.1.7 Závěr analýzy

„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."[18]. Ke stejným závěrům lze dojít i pro tento kolejní klub Sincoolka. Systém pro kolejní klub Sincoolka by měl splňovat unikátní funkční požadavky, jako je například přidělování kelímků, možnost využít tiskárnu za odpovídající poplatek a mnoho dalšího. Systém by neměl obsahovat základní bezpečnostní zranitelnosti. Nový vývojář neznalý systému by se v něm měl lehce zorientovat a nebýt znechucen stavem kódu. V neposlední řadě by měl být systém jednoduše nasaditelný. Tyto požadavky nesplňuje bohužel žádný systém a případné úpravy existujících systémů se zdají býti více časově náročné než implementace nového řešení.

3.2 Nefunkční požadavky

Nefunkční požadavky představují specifikaci očekávané funkcionality systému, která se nezaměřuje na konkrétní funkční vlastnosti, ale spíše na aspekty zajišťující správné a efektivní fungování systému.

N1: Kód musí být přehledný

Popis: Přehlednost bude zaručena rozdělením systému do modulů.

N2: Autorizace a autentizace

Popis: Systém bude autorizovat a autentizovat uživatele na základě prokázání znalosti hesla, nebo tokenu.

N3: REST API

Popis: Serverová část bude implementovat REST API [21].

N4: Framework

Popis: Aplikační vrstva bude implementována pomocí frameworku, konkrétně se bude jednat o Spring.

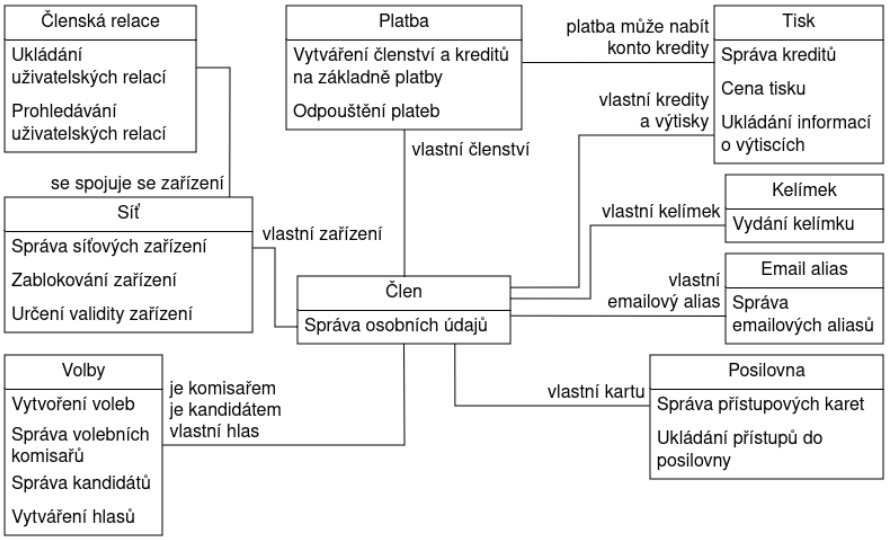
3.3 Funkční požadavky

Funkční požadavky jsou specifikací toho, co systém musí dělat. Tyto požadavky definují, jakým způsobem systém reaguje na uživatelské vstupy, zpracovává data a provádí různé operace.

Funkční požadavky vznikly z funkčních požadavků předchozího systému a z požadavků rozšíření systému. Požadavky byly zkonzultovány s Bc. Pavlem Valachem, který je hlavním správcem současného řešení. Pro lepší pochopení některých funkčních požadavků je v jejich popisech naznačen související business proces.

Implementace současné informačního systému pro správu kolejního klubu Sincoolka je nepřehledná. Přehlednost se dá zvýšit rozdělením aplikace na moduly. Vzhledem k tomu, že se jedná

o známý nedostatek, už v analýze se budou funkční požadavky dělit do modulů. Schéma modulů je zobrazeno pomocí diagramu 3.1.



Obrázek 3.1 Schéma modulů

MOSCoW je technika prioritizace požadavků. Tato klasifikace pomáhá lépe porozumět, které požadavky jsou důležité, a které nikoliv. [22]. Rozděluje požadavky do čtyř kategorií:

1: Must have

Překlad: Musí mít

Zkratka: M

2: Should have

Překlad: Měl by mít

Zkratka: S

3: Could have

Překlad: Mohl by mít

Zkratka: C

4: Won't have

Překlad: Nemusí mít

Zkratka: W

3.3.1 Election – Volby

Na koleji se konají volby, volí se do zastupitelstva Sinkuleho a Dejvické koleje nebo představenstva klubu Sincoolka. Volit se dá pomocí urny a on-line pomocí systému. Je nutné zajistit 100% anonymitu, žádný hlas nepůjde zpětně spojit s kandidátem, pro kterého bylo hlasováno, protože správci systému se občas stávají kandidáty a případné zpětné odhalení, kdo pro koho volil by mohlo způsobit špatnou pověst voleb a toxickou atmosféru v klubovém vedení. Navíc je anonymita hlasování standardním prvkem a požadavkem voleb.

F1: Vytvoření voleb

Popis: Systém umožňuje vytvořit volby. Je nutné rozlišit typy voleb, volí se do zastupitelstva Sinkuleho a Dejvické koleje nebo představenstva klubu Sincoolka. Dále je vhodné definovat časové rozmezí pro příjem kandidátek a hlasů. K volbám půjde následně přiřadit členy volební komise. Průběh několika voleb může mít časové kolize, neboli volí se zároveň do představenstva i zastupitelstva.

Priorita: S

Složitost: Střední

Je ve starém systému: Ne

F2: Přidání a smazání členů volební komise

Popis: Lze přidat nebo odebrat členovi jeho účast ve volební komisi. Veškerá přidání jsou zaznamenána, konkrétně se zaznamená, kdo a kdy je provedl.

Priorita: S

Složitost: Střední

Je ve starém systému: Ne

F3: Příjem kandidátek

Popis: Člen klubu se může stát kandidátem. Musí kontaktovat volební komisi a předat jim volební plakát. Volební komise následně zapíše kandidáta do systému i s jeho plakátem neboli kandidátkou. Registrovat kandidátky lze jen v určitém období, nastaveném při vzniku voleb. Volební plakát je jednostránkový dokument typu PDF.

Priorita: S

Složitost: Střední

Je ve starém systému: Ano

F4: Hlasování

Popis: Každý člen může hlasovat jen jednou. Lze hlasovat elektronicky i fyzicky za pomoci urny. O urnové volby se stará volební komise, která označuje a ověřuje členy, kteří hlasovali. O elektronické volby se stará systém, který je sám zahájí i ukončí.

Priorita: M

Složitost: Vysoká

Je ve starém systému: Ano

F5: Vyhodnocení voleb

Popis: Probíhá po skončení hlasování. Do systému se zadají urnové hlasy. Systém spočítá hranici pro zvolení a vyhodnotí, zdali je kandidát zvolen. Následně sestaví výsledné pořadí kandidátů. V případě stejného počtu hlasů lze upravit pořadí kandidátů ručně.

Priorita: C

Složitost: Vysoká

Je ve starém systému: Ne

3.3.2 Email Alias – Emailový alias

Systém umožňuje členům přidávat email aliasy.

F6: Přidání emailového aliasu

Popis: Člen si může přidat emailový alias, pokud už není používán někým jiným. Alias slouží k příjemnější a snazší identifikaci člena v rámci elektronické komunikace. Výchozí alias je vytvořen při registraci.

Priorita: S

Složitost: Nízká

Je ve starém systému: Ano

F7: Smazání emailového aliasu

Popis: Člen může smazat svůj emailový alias, pokud již o daný alias nemá zájem. Člen musí mít vždy alespoň jeden aktivní emailový alias. Smazání aliasu umožňuje udržovat čistotu a efektivitu správy elektronické pošty.

Priorita: S

Složitost: Nízká

Je ve starém systému: Ano

3.3.3 Fitness – Posilovna

Kolejní klub poskytuje posilovnu pro členy, kteří si zaplatili fitness členství. Především je nutné vědět, jaké členy lze vpustit do posilovny. Dále je vhodné mít informace, kdo a kdy posilovnu navštívil v případě vyskytnutí problému, například krádeže.

F8: Přístup do posilovny

Popis: Člen přiloží kartu ke čtečce, která kartu načte a ověří. Pokud je karta přiložena v návštěvní dobu, je členovi povolen vstup do posilovny. Správce posilovny má přístup kdykoliv. Systém Sinis poskytne rozhraní pro výpis všech platných karet. Platnost karty člena je určena absencí zablokování přístupu do posilovny a platným fitness členstvím. Výpis musí obsahovat informace o tom, zda je člen správcem posilovny, či nikoliv.

Priorita: M

Složitost: Vysoká

Je ve starém systému: Ano

F9: Registrace karty

Popis: Správce posilovny zaregistruje kartu členovi. Každý člen může mít pouze jednu aktivní kartu.

Priorita: M

Složitost: Nízká

Je ve starém systému: Ano

F10: Odebrání karty

Popis: Systém umožňuje smazání karty. Je nutné vědět, kdo a kdy a proč kartu smazal.

Priorita: S

Složitost: Nízká

Je ve starém systému: Ano

F11: Uložení vstupu do posilovny

Popis: Systém umožňuje ukládat vstupy do posilovny a snaží se je spojit s příslušnou kartou, kterou vlastní člen.

Priorita: M

Složitost: Střední

Je ve starém systému: Ano

F12: Procházení vstupů

Popis: Správce posilovny má možnost procházet vstupy do posilovny.

Priorita: M

Složitost: Nízká

Je ve starém systému: Ano

3.3.4 Member – Člen

Tento model slouží především pro správu osobních údajů člena.

F13: Registrace člena

Popis: Zájemce o členství může vyplnit před-registrační formulář a následně přijít do LABu, kde bude ověřena jeho identita registrátorem a vyplněna fyzická smlouva, kterou žadatel podepíše. Pokud nejsou vyplněny před-registrační údaje, lze je doplnit na místě za pomoci registrátora. Člen je registrován a uložen v systému. Členství je plně aktivováno až po zaplacení členských příspěvků. Systém především realizuje před-registrační formulář a uložení člena. Je třeba znát celé jméno, místo a datum narození, trvalou adresu, číslo pokoje a univerzitu a fakultu, na které studuje. Nutnou podmínkou pro členství není studium na vysoké škole ani ubytování na pokojích Sinkuleho a Dejvické koleje.

Priorita: M

Složitost: Vyšší

Je ve starém systému: Ano

F14: Změna hesla

Popis: Člen požádá o změnu hesla, pověřená osoba ověří jeho identitu a následně mu umožní změnu hesla. Člen si může změnit heslo sám v systému po prokázání znalosti aktuálního hesla.

Priorita: S

Složitost: Střední

Je ve starém systému: Ano

F15: Zrušení účtu

Popis: Pokud člen podal žádost o zrušení účtu, má klub zákonnou povinnosti smazat jeho údaje do konce následujícího roku. Žádostí o zrušení účtu člen automaticky přestává mít výhody členství. Zrušením účtu dochází k deaktivaci veškerých členských služeb, především přístupu k síti a do posilovny. Kvůli poskytování internetu je zákonná povinnost uchovávat údaje minimálně na 6 měsíců. Aby bylo umožněno zpětné dohledání majitele uživatelské relace.

Priorita: C

Složitost: Střední

Je ve starém systému: Ne

F16: Přidání a smazání adresy

Popis: Systém umožňuje přidat adresu členovi klubu nebo ji smazat. Systém ukládá i smazané adresy.

Priorita: C

Složitost: Nízká

Je ve starém systému: Ano, ale nejde smazat.

F17: Vytvoření země

Popis: Registrátor může do systému přidat novou zemi, za účelem vytvoření adresy, či místa narození budoucího nebo současného člena.

Priorita: C

Složitost: Nízká

Je ve starém systému: Ne

F18: Změna pokoje na koleji

Popis: Pokud se člen přestěhuje na jiný pokoj, zajde do LABu, kde mu registrátor smaže původní pokoj a vytvoří nový. Podle pokoje se určuje možnost účasti ve volbách, kde se volí zástupce koleje. Systém ukládá i smazané pokoje.

Priorita: S

Složitost: Střední

Je ve starém systému: Ano

F19: Přidání a odebrání univerzity a fakulty

Popis: Člen nebo registrátor může přidávat a odebírat spojení mezi členem a fakultou. V systému budou uloženy veškeré změny.

Priorita: S

Složitost: Střední

Je ve starém systému: Ano

F20: Vytvoření univerzity a fakulty

Popis: Registrátor může vytvořit v systému fakultu nebo univerzitu.

Priorita: C

Složitost: Nízká

Je ve starém systému: Ne

F21: Správa rolí

Popis: Oprávněná osoba na základě podnětu může přidávat a odebírat role členům. Vždy musí mít vyšší oprávnění než přidávaná a odebíraná role. Předseda se stará o hlavní správce posilovny a hlavní správce sítě, dále spravuje pokladníky a správce kelímků. Hlavní správcové posilovny a sítě se starají o správce posilovny a sítě. Je vhodné ukládat historii kdo a kdy byl jakým správcem.

Priorita: S

Složitost: Střední

Je ve starém systému: Ano

F22: Zablokování přístupu

Popis: Správce může zablokovat členovi přístup na určitou dobu. Lze zakázat přístup do posilovny a připojení k internetu. Internet odstřihávají správci sítě a přístup do posilovny zablokovávají správci posilovny. Každé zablokování musí mít důvod, který bude přiložen k zablokování ve formě textové zprávy.

Priorita: M

Složitost: Střední

Je ve starém systému: Ano, ale implementace ve starém systému je podivná.

3.3.5 Network – Síť

Hlavní funkcionalitou je správa zařízení, které se připojí k internetu.

F23: Přidání zařízení pro připojení k internetu

Popis: Systém umožňuje přidat zařízení pro připojení k internetu. Každé zařízení musí mít unikátní MAC adresu. Je vhodné přidat popis, aby přidaná zařízení šla jednoduše rozeznat.

Priorita: M

Složitost: Střední

Je ve starém systému: Ano

F24: Smazání zařízení pro připojení k internetu

Popis: Registrátor nebo člen může smazat zařízení pro připojení k internetu. Člen může smazat pouze svoje zařízení. Systém ukládá i smazané zařízení a k němu informace kdo, kdy a proč dané zařízení smazal.

Priorita: M

Složitost: Nízká

Je ve starém systému: Ano

F25: Vypsat všechna validní zařízení

Popis: Systém zobrazí všechna zařízení, která mají přístup k internetu. Zařízení nesmí být zablokované ani smazané a jeho majitel by měl být držitelem platného členství. Platné členství nemusí mít uživatelé s čerstvě ověřenou identitou, nebo ti, kteří nedávno provedli před-registraci.

Priorita: M

Složitost: Vysoká

Je ve starém systému: Ano

F26: Zablokování zařízení

Popis: Správce sítě může zablokovat konkrétní zařízení. Zařízení se blokuje na určitou dobu. Systém ukládá kdo, kdy a proč zařízení zablokoval a umožňuje zablokování smazat.

Priorita: M

Složitost: Střední

Je ve starém systému: Ne

F27: Správa switch

Popis: Správce sítě může přidat a odebrat switch.

Priorita: C

Složitost: Střední

Je ve starém systému: Ne

F28: Správa socketu

Popis: Správce sítě může přidat a odebrat socket. Sockety jsou umístěné v pokojích.

Priorita: C

Složitost: Střední

Je ve starém systému: Ne

F29: Správa portu

Popis: Správce sítě může přidat a odebrat port. Port propojuje socket a switch. S portem se budou spojovat uživatelské relace.

Priorita: C

Složitost: Střední

Je ve starém systému: Ne

3.3.6 Payment – Platba

Hlavním úkolem tohoto modulu je automatizovat mapování plateb. Platby by měly vytvářet korespondující členství nebo nabít konto sloužící pro tisk na tiskárně.

F30: Vytvoření platby

Popis: Aplikace umožňuje vytvořit platbu, která vznikla bankovním převodem nebo pomocí platby kartou přes platební bránu. Každou platbu se systém pokusí spárovat se členem a vytvořit členství nebo nabít konto kredity.

Priorita: C

Složitost: Střední

Je ve starém systému: Ano

F31: Dodatečné spárování

Popis: Pokladník na základě podnětu od člena páruje nespárovanou platbu. Platba se mapuje buďto na členství nebo na tisk. V případě nesprávného automatického i ručního spárování je možné párované platby zrušit a vytvořit nové. Systém ukládá informace o tom, jak párování vzniklo, případně informace o tom kdo, kdy a proč provedl smazání párování.

Priorita: M

Složitost: Vysoká

Je ve starém systému: Ano

F32: Odpuštění platby

Popis: Pokladník může odpustit platbu za členství, jak základního tak i sportovního. Obvykle se bude odpouštět platba aktivním členům, kteří se starají o chod klubu.

Priorita: M

Složitost: Střední

Je ve starém systému: Ano

F33: Vytvoření semestru

Popis: Pokladník má právo vytvářet semestr, na nějž se budou vázat členství a tím pádem mapovat platby. Každý semestr někdy začíná a končí, ale mapování plateb striktně neodpovídá začátku a konci semestru.

Priorita: M

Složitost: Střední

Je ve starém systému: Ano

F34: Upravení semestru

Popis: Pokladník má právo upravovat semestr.

Priorita: S

Složitost: Střední

Je ve starém systému: Ne

F35: Prohlížení plateb

Popis: Člen má možnost si prohlédnout své spárované platby.

Priorita: S

Složitost: Nízká

Je ve starém systému: Ano

F36: Prohlížení členství

Popis: Správci mají možnost prohlížet členství ostatních členů. Člen si může prohlížet pouze vlastní členství.

Priorita: S

Složitost: Nízká

Je ve starém systému: Ano

3.3.7 Print – Tisk

Modul umožňuje za poplatek vytisknout dokument.

F37: Nabití konta offline

Popis: Člen si dojde do LABu v úřední hodiny a správce sítě nebo pokladník mu smění hotovost za kredity, případně vytvoří daňový doklad. Systém pouze eviduje transakce kreditů. Ukládá, kdo a kdy transakci vytvořil případně smazal.

Priorita: M

Složitost: Střední

Je ve starém systému: Ano

F38: Nabití konta převodem

Popis: Kredity si lze nabít i on-line, buďto pomocí platby kartou, nebo bankovním převodem. Pokud člen vyplní správné atributy platby, systém by měl automaticky vytvořit patřičnou transakci.

Priorita: S

Složitost: Střední

Je ve starém systému: Ne

F39: Nabití bonusových kreditů

Popis: Bonusové kredity přiděluje správce sítě nebo pokladník, pokud se výtisk týká provozu kolejního klubu.

Priorita: S

Složitost: Střední

Je ve starém systému: Ano

F40: Tisk

Popis: Člen si požádá o tisk, systém mu odečte patřičné kredity a v případě nedostatku kreditů zobrazí chybovou zprávu. Jako první se odečtou bonusové kredity. Systém ukládá informace o tisku, konkrétně počet vytisknutích stran a spotřebu toneru.

Priorita: S

Složitost: Vysoká

Je ve starém systému: Ano

F41: Reklamace

Popis: Výtisk se nepovede. Člen donese špatný výtisk do LABu a požádá o vrácení peněz. Pověřená osoba ověří, že reklamace je validní a vrátí členovi jeho kredity.

Priorita: S

Složitost: Střední

Je ve starém systému: Ano

F42: Zrušení konta / Vrácení peněz

Popis: Pokud se člen rozhodne směnit kredity na hotovost musí zajít do LABu, kde mu kredity budou směněny na hotovost. Systém uloží informaci o směně kreditů na hotovost.

Priorita: S

Složitost: Střední

Je ve starém systému: Ano

F43: Změna ceny tiskařských potřeb

Popis: Pokladník může změnit cenu toneru nebo papíru. V systému se ukládá od kdy do kdy je cena platná, kdo a kdy cenu nastavil. Případnou změnu ceny lze smazat.

Priorita: S

Složitost: Vysoká

Je ve starém systému: Ne

F44: Historie ceny zdrojů pro tisk

Popis: Pokladník může procházet historii cen zdrojů pro tisk.

Priorita: S

Složitost: Nízká

Je ve starém systému: Ne

3.3.8 Session – Uživatelské relace

Tento modul ukládá členské relace a snaží je spojit s portem a zařízením.

F45: Zaznamenávání uživatelských relací

Popis: Systém bude ukládat uživatelské relace a pokoušet se je spojit s portem a se zařízením pomocí MAC adresy. Jedna uživatelská relace může obsahovat několik ip adres.

Priorita: S

Složitost: Střední

Je ve starém systému: Ne

F46: Vyhledání uživatelských relací

Popis: Správce sítě může prohledávat uživatelské relace.

Priorita: S

Složitost: Střední

Je ve starém systému: Ne

3.3.9 Stuff – Kelímek

Tento modul umožňuje rozdávat členům klubu kelímky. Správa kelímků by měla být rozšiřitelná, například pro distribuci jiných předmětů.

F47: Vydání kelímku

Popis: Systém ukládá komu byly kelímky vydány a komu ne. Dále jsou zaznamenány informace, kdo a kdy kelímek vydal. Člen má nárok na jeden kelímek za život.

Priorita: S

Složitost: Střední

Je ve starém systému: Ano

F48: Zrušení vydání kelímku

Popis: Systém umožňuje zrušení vydání kelímku, v případě závady na vydaném kelímku, či z jiného rozumného důvodu. Je nutné vědět kdo, kdy a proč zrušil vydání kelímku.

Priorita: S

Složitost: Střední

Je ve starém systému: Ano

3.4 Aktéři

1: Zájemce o členství

Popis: Zájemce o členství je uživatel, který nemá účet a není členem klubu. V systému však může vyplnit před-registrační formulář pro zrychlení procesu registrace.

2: Člen klubu

Popis: Člen klubu je uživatel, který je plnohodnotným členem. Má ověřenou identitu a nepožádal o smazání účtu. Zároveň je držitelem platného základního členství, případně i sportovního. Člen může spravovat vlastní účet, prohlížet si informace v systému týkající se jeho osoby a účastnit se voleb.

3: Registrátor

Popis: Registrátor je členem klubu a navíc pomáhá s registrací nových členům, především ověřuje jejich identitu. Má právo přidávat a mazat členovi síťová zařízení. Registrátor může obsloužit žádost o smazání účtu. Registrátorem se člen stane obvykle na vlastní žádost.

4: Správce sítě

Popis: Správce sítě dědí práva registrátora a má navíc práva pro správu sítě, kterými jsou přidávání a mazání soketů, portů a přepínačů. Dále může zablokovat konkrétní zařízení, nebo členovi úplně zablokovat přístup k internetu. Také spravuje uživatelské relace.

5: Hlavní správce sítě

Popis: Má veškerá práva jako správce sítě. Navíc povyšuje členy na registrátory a správce sítě. Může i odebírat role registrátor a správce sítě.

6: Pokladník

Popis: Pokladník se především stará o párování plateb, odpouštění plateb správcům. Nabíjení konta kredity pro tisk. Jedná se samozřejmě o člena klubu. Navíc má práva pro změnu ceny zdrojů pro tisk, jako je například papír a toner.

7: Správce posilovny

Popis: Správce posilovny je členem klubu a má práva pro spravování posilovny, jedná se především o přidávání vstupních karet.

8: Hlavní správce posilovny

Popis: Má veškerá práva jako správce posilovny. Navíc povyšuje členy na správce posilovny, také může správce posilovny degradovat na prostého člena.

9: Správce kelímků

Popis: Vydává kelímky členům a přijímá případné reklamace.

10: Předseda

Popis: Předseda přidává a odebírá hlavní správce sítě a hlavní správce posilovny, dále vytváří členy volební komise.

11: Člen volební komise – komisař

Popis: Člen volební komise může přidat nového člena do volební komise a odebrat stávajícího. Volební komise přijímá kandidáty a přidává součet urnových hlasů, také ověřuje a označuje voliče, kteří volili v urnových volbách. Volební komise je vázána pouze na jedny volby. Člen může být členem volební komise několika po sobě jdoucích voleb. Komisař nesmí být kandidátem.

12: Správce všeho – God admin

Popis: Má práva všech správců.

3.5 Případy užití

Případy užití jsou scénáře, které popisují, jak uživatelé interagují se systémem. Případy užití jsou vyobrazeny pomocí diagramu A.9.

UC1: Správa osobních údajů

Aktér: Člen

Scénář: Člen si vybere údaj, který chce upravit. Může smazat stávající nebo přidat nový. Mezi osobní údaje patří emailový alias, síťové zařízení, fakulta a adresa. Po odeslání na server se data uloží.

UC2: Vytvoření hlasu ve volbách

Aktér: Člen

Popis: Člen vybere volby, ve kterých chce volit. Vybere kandidáta nebo kandidáty a odešle hlas. Systém označí člena jako voliče, který již odevzdal hlas a kandidátovy zvýší počet hlasů.

UC3: Vytvoření výtisku

Aktér: Člen

Popis: Člen vytiskne dokument na tiskárně a systém odečte kredity z konta v závislosti na využití toneru a počtu stran.

UC4: Správa síťových zařízení

Aktér: Registrátor a člen

Popis: Registrátor vybere člena a vybere síťové zařízení, které chce smazat. Vybranému nebo členovi bude moci také vytvořit nové zařízení. Člen si může upravovat vlastní síťová zařízení.

UC5: Registrace uživatele

Aktér: Registrátor

Popis: Registrátor si zobrazí před-registrační formulář žadatele, pokud existuje. Zkontroluje žadateli identitu. Zkontroluje, doplní či vytvoří nutné údaje k registraci a zaregistruje uživatele. Systém uloží nového člena.

UC6: Požádání o smazání účtu

Aktér: Registrátor

Popis: Člen může požádat o smazání účtu. Registrátor najde členský účet a požádá o jeho smazání. Systém žádost uloží a účet smaže dle zákonných pravidel.

UC7: Zablokování internetového připojení

Aktér: Správce sítě

Popis: Správce sítě vybere člena, kterému následně zablokuje na určitou dobu připojení k internetu. Systém zablokování uloží a po dobu zablokování nebude zařízení vlastněná členem zobrazovat ve validních zařízeních.

UC8: Správa prvků sítě

Aktér: Správce sítě

Popis: Správce přidá nový prvek sítě, konkrétně přepínač, port nebo soket. Pokud vybere existující prvek tak ho může i smazat. Systém ukládá změny.

UC9: Zablokování internetového zařízení

Aktér: Správce sítě

Popis: Správce sítě vybere zařízení, které následně zablokuje na určitou dobu, dle jeho uvážení. Systém zablokování uloží a po dobu zablokování nebude zablokované zařízení zobrazovat ve validních zařízeních.

UC10: Přidání uživatelské relace

Aktér: Správce sítě

Popis: Správce sítě může přidat uživatelskou relaci. Systém se ji pokusí spojit se zařízením a s portem.

UC11: Prohledávání uživatelských relací

Aktér: Správce sítě

Popis: Správce sítě může prohledávat a filtrovat uživatelské relace.

UC12: Správa správců sítě

Aktér: Hlavní správce sítě

Popis: Vybere člena a přidá mu či odebere roli registrátora nebo správce sítě. Systém uloží změny.

UC13: Správa hlavních správců

Aktér: Předseda

Popis: Vybere člena a přidá mu či odebere roli hlavního správce sítě nebo hlavního správce posilovny. Systém uloží změny.

UC14: Vytvoření voleb

Aktér: Předseda

Popis: Může vytvořit volby, vybere jakého jsou typu a nastaví další volební parametry. Systém uloží nové volby.

UC15: Správa volebních komisařů

Aktér: Volební komisař a Předseda

Popis: Vybere člena a přidá mu, nebo odebere členství ve volební komisi. Systém uloží změny.

UC16: Vyplnění před-registračního formuláře

Aktér: Zájemce o členství

Popis: Člen vyplní formulář konkrétně jméno, fakultu, pokoj, adresu, místo a datum narození. Zájemce data odešle a systém data uloží.

UC17: Párování plateb

Aktér: Pokladník

Popis: Vybere platbu a následně ji může namapovat na typ členství, semestr a člena. Následně data odešle. Systém tím pádem vytvoří členství s příslušnými vazbami. Nebo může platbu spojit s tiskem a systém vytvoří transakci, která nabije členovi konto patřičnými kredity.

UC18: Odpouštění plateb

Aktér: Pokladník

Popis: Vybere člena a vytvoří mu odpustek s odůvodněním. Systém ukládá kdo, kdy, proč a komu byl odpuštěn poplatek za členství.

UC19: Změna ceny tisku

Aktér: Pokladník

Popis: Pokladník změní cenu papíru nebo toneru do tiskárny. Systém změny uloží a bude je při výpočtech ceny tisku respektovat.

UC20: Správa konta

Aktér: Pokladník a Správce sítě

Popis: Vyberou člena a nahrají mu kredity, lze nahrát i zápornou hodnotu v případě zrušení konta. K transakci musí přidat komentář. Systém navíc uloží autora transakce.

UC21: Vydání kelímku

Aktér: Správce kelímků

Popis: Vyhledá člena a přiřadí mu kelímek, či zruší vydání kelímku. Systém uloží nový stav i s autorem změny.

UC22: Správa přístupových karet

Aktér: Správce posilovny

Popis: Vyhledá člena a přidá mu přístupovou kartu do posilovny nebo odebere stávající. Systém uloží změny a bude je reflektovat, pokud bude požádán o výpis platných karet.

UC23: Správa správců posilovny

Aktér: Hlavní správce posilovny

Popis: Vybere člena a přidá mu či odebere roli správce posilovny. Systém uloží změny.

UC24: Přidávání offline hlasů

Aktér: Volební komisař

Popis: Sečte hlasy z urny a následně je nahraje do systému. Systém pomocí nich bude počítat vítěze voleb.

UC25: Správa kandidátů

Aktér: Volební komisař

Popis: Člen zažádá o kandidaturu. Komisař označí člena za kandidáta a nahraje jeho plakát. Na žádost člena lze kandidaturu zrušit. Systém ukládá kandidáty i s jejich plakáty.

3.6 Diagram aktivit

Nejsložitější aktivita, která se objevuje v klubu Sincoolka je registrace nového člena, zobrazená pomocí diagramu A.10. Zbylé aktivity jsou přímočaré, či naprosto intuitivní. Cílem registrace člena je získat nutné údaje pro jednoznačnou identifikaci osoby. Informace se uchovávají, protože kolejní klub poskytuje internetové připojení. Poskytovatelé internetu mají zákonnou povinnost ukládat informace o uživatelích. S ohledem na časovou náročnost procesu získání potřebných informací pro registraci nového uživatele a vzhledem k tomu, že většina nových členů se registruje na začátku školního roku, byla přidána možnost vyplnění před-registračního formuláře. Tento krok má za cíl urychlit a zjednodušit celý proces registrace.

Kapitola 4

Návrh nového kolejního systému

Tato kapitola se věnuje návrhu nového kolejního systému. Jsou zde popsány technologie, struktura systému a diagram stavů pro platbu kartou.

4.1 Technologie nového kolejního systému

Systém bude v budoucnosti spravován dalšími dobrovolníky, kteří nebyli přítomni při počátcích vývoje. Hlavní požadavkem na použité technologie bude jejich popularita, která zajistí, že dobrovolník bude mít s technologií zkušenost či zájem se ji naučit. Dalším požadavkem bude široká podpora, která je lehce spojena s popularitou. Technologie nesmí mít bezpečnostní zranitelnosti. Ideální bude, když technologie bude mít komunitu, která ji stále aktivně vyvíjí.

4.1.1 Spring

Spring [19] je framework nejenom pro jazyk Java, který zrychluje, zjednodušuje a zajišťuje bezpečnost programování. Jeho zaměření na rychlost, jednoduchost a produktivitu ho činí nejpopulárnějším Java frameworkem na světě. Díky svým flexibilním knihovnám je Spring důvěrně známý mezi vývojáři po celém světě. Poskytuje příjemné zkušenosti milionům koncových uživatelů, ať už jde o streamování TV, on-line nakupování nebo jiná řešení. Spring také získal příspěvky od všech velkých jmen v technologickém průmyslu, jako je například Alibaba, Amazon, Google, Microsoft a dalších. Díky flexibilnímu a komplexnímu souboru rozšíření a knihoven třetích stran umožňuje Spring vývojářům vytvářet téměř jakoukoli aplikaci.

Pro tento systém byl Spring zvolen, protože splňuje veškeré náležitosti definované v úvodu této sekce. S jeho pomocí lze implementovat funkční i nefunkční požadavky. Jednou z dalších výhod je, že se vyučuje na ČVUT FIT, což může motivovat člena klubu ke správě tohoto systému. Spring je distribuován pod licencí Apache License 2.0, což znamená, že je k dispozici zdarma pro nekomerční i komerční projekty, což je další pozitivní aspekt frameworku Spring. [19]

4.1.2 Java

Programovací jazyk se rozhodoval mezi Javou a Kotlinem [23]. Oba jazyky využívají JVM [24] a jsou podporovány frameworkem Spring. Java i Kotlin jsou schopné splnit funkční požadavky systému. Kotlin je novější a plně interoperabilní s Javou, což lze považovat v našem kontextu za nevýhodu, pokud se vezme v potaz stav současného systému, tak je konzistence velkým problémem, z čehož lze usuzovat, že pokud bude zvolen jazyk Kotlin, bude při budoucím přidáváním

nových požadavků polovina souborů napsána v čisté Javě a polovina v Kotlinu, v horším případě se syntaxe obou jazyků potká v jednom souboru.

Kotlin nabízí úspornější a elegantnější syntaxi oproti Javě. Z hlediska popularity je na tom Kotlin jednoznačně hůře než Java, ale oba mají aktivní komunity, které se starají o vývoj. Java je signifikantně více rozšířená, má větší podporu a Spring framework je primárně napsaný pro ni, proto je Java zvolena pro vývoj tohoto projektu. [23]

Java [9] je vysoko úrovňový jazyk s objektově orientovaným přístupem, s automatickým řízením paměti, aby se zabránilo bezpečnostním problémům explicitní dealokace. Java patří mezi silně a staticky typované jazyky. Při návrhu se přihlíželo k jednoduchosti, za účelem snadného chápání kódu. Je příbuzná s jazyky C a C++, ale je organizovaná poněkud odlišně, přičemž některé prvky C a C++ jsou vynechány a některé myšlenky z jiných jazyků jsou zahrnuty.

4.1.3 PostgreSQL

PostgreSQL [6] je relační databázový systém s důrazem na rozšiřitelnost a spolehlivost. Hlavní požadavky na databázový systém bylo, aby byl relační a zadarmo. PostgreSQL tyto požadavky splňuje. Navíc je použit v současném řešení, což lze vnímat jako další pozitivum, protože současní správci nebudou mít problém s přechodem na novou technologii. Podrobnější popis se nalézá v podsekci sekci 2.1.4.

4.1.4 Gradle

Gradle Build Tool [25] je rychlý, spolehlivý a přizpůsobivý nástroj pro automatizaci sestavování pomocí elegantního, rozšiřitelného a deklarativního jazyka. Gradle je široce používaný nástroj s aktivní komunitou a silným ekosystémem vývojářů. Gradle podporuje Android, Java, Kotlin Multiplatform, Groovy, Scala, JavaScript a C/C++. Všechny hlavní vývojové prostředí podporují Gradle, včetně Android Studia, IntelliJ IDEA, Visual Studio Code, Eclipse a NetBeans.

Pro sestavení projektu byla možnost si vybrat z několika nástrojů. Finální volba padla na nástroj Gradle. Hlavním důvodem volby je jeho rostoucí popularita a široká podpora. Oproti svému hlavnímu konkurentu se jménem Maven má hezčí syntaxi a je rychlejší. [26]

4.1.5 Docker Compose

Docker Compose [27] je nástroj pro definování a spouštění multi-kontejnerových aplikací. Slouží k jednoduchému a efektivnímu vývoji a nasazení. Compose zjednodušuje kontrolu celého aplikačního stacku, umožňuje snadné spravování služeb, sítí a svazků v jednom srozumitelném konfiguračním souboru typu YAML. Poté s jediným příkazem lze vytvořit a spustit všechny služby z konfiguračního souboru. Compose funguje ve všech prostředích: produkčním, vývojovém, testovacím.

Konfigurační soubory Docker Compose jsou snadno sdílitelné, což usnadňuje spolupráci mezi vývojáři, operačními týmy a dalšími zainteresovanými stranami. Tento spolupracující přístup vede k hladším pracovním postupům, rychlejšímu řešení problémů a zvýšení celkové efektivity. [28]

Docker je využit v této práci pro spouštění PostgreSQL databáze. Soubor YAML použitý v této práci lze vidět v ukázce kódu 4.1.

```yaml
version: '3.7'
services:
  postgres:
    image: postgres:15
    restart: always
    environment:
      - POSTGRES_USER=sinis
      - POSTGRES_PASSWORD=sinis
      - POSTGRES_DB=sinis_db
    ports:
      - '5432:5432'
    volumes:
      - .docker/postgres-data:/var/lib/postgresql/data
```

Výpis kódu 4.1 Ukázka kódu ze souboru docker-compose.yml

4.1.6 Spring Data JPA

Pro mapování dat do databáze se používá Spring Data JPA [29]. Jedná se o součást rozsáhlejší rodiny Spring Data a umožňuje snadnou implementaci repositářů založených na JPA. JPA je zkratkou pro Java Persistence API.

Zajištění datového přístupu pro aplikaci může být náročný úkol. Je třeba psát příliš mnoho kódu i pro nejjednodušší dotazy. Spring Data JPA si klade za cíl významně zlepšit implementaci datové vrstvy. [29]

4.1.7 Springdoc

Pro generování dokumentace byla zvolena knihovna springdoc-openapi [30]. Jedná se o java knihovnu, která pomáhá automatizovat generování dokumentace API ve Spring projektech. Springdoc-openapi funguje tak, že zkoumá aplikaci za běhu, aby odvodila sémantiku API na základě Spring konfigurací, struktury tříd a různých anotací. Podporuje automatické generovaní dokumentací ve formátu JSON/YAML a HTML. Tuto dokumentaci lze doplnit komentáři pomocí anotací swagger-api.

4.1.8 Postman

Postman [31] je platforma pro tvorbu a používání API, která zjednodušuje každý krok životního cyklu API a usnadňuje spolupráci pro rychlejší vytváření API. Umožňuje ukládání, katalogizaci a spolupráci kolem všech API artefaktů na jedné centrální platformě, včetně specifikací, dokumentace, testovacích případů a metrik. S kompletním souborem nástrojů pro návrh, testování, dokumentaci a sdílení API urychluje životní cyklus API. Postman podporuje plný životní cyklus řízení API, umožňuje přesun k lepší kvalitě API a podporuje spolupráci mezi týmy. Pracovní postupy pomáhají organizovat práci s API a spolupracovat napříč organizací. Integrace s klíčovými nástroji ve vývojové pipeline, rozšiřitelnost skrze Postman API a otevřené technologie zajišťují flexibilitu a rozšíření možností. Postman je dostupný zdarma s možností přechodu na placené plány pro pokročilé možnosti.

Postman byl vybrán především kvůli jeho intuitivnímu použití, popularitě a správci současného systému ho sami navrhli. Endpointy v aplikaci Postman navíc usnadní vývoj klientské části

aplikace, protože si budoucí vývojář může endpointy procházet a interagovat s nimi, posílat data a analyzovat odpovědi, což je rozhodně lepší než statická dokumentace. Další výhodou aplikace Postman je její nabídka předdefinovaných funkcí pro testování a možnosti vytvoření náhodných dat, která se vkládají do těla žádosti anglicky zvaného requet.

4.1.9 JavaScript

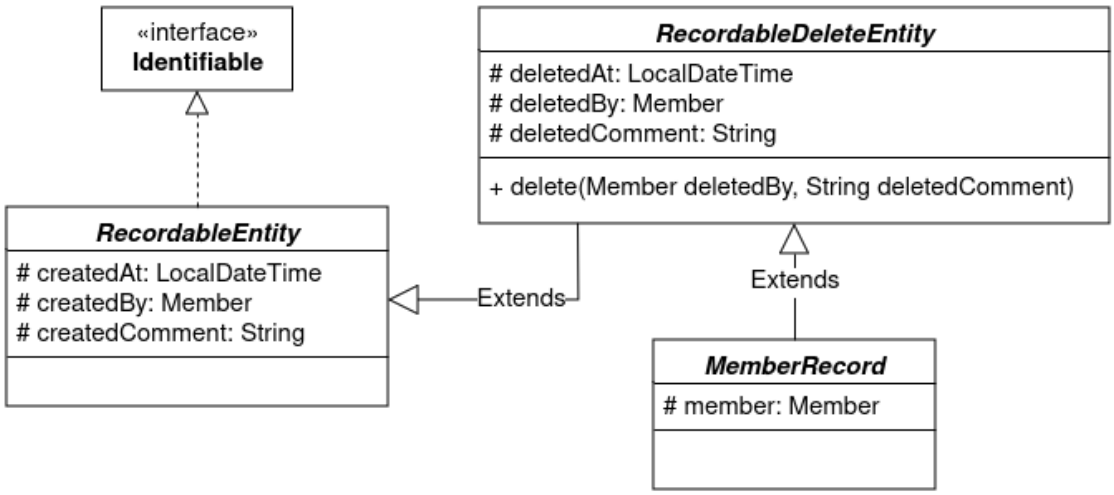
Testování v aplikaci Postman je implementováno pomocí jazyku JavaScript. JavaScript byl zvolen, protože se jedná o výchozí a nejpoužívanější volbu pro Postman [32]. JavaScript byl již popsán v kapitole zabývající se současným softwarovým řešením, konkrétně v sekci 2.1.5.

4.2 Struktura

Nový informační systém je rozdělen do několika modulů: volby, emailové aliasy, posilovna, správa osobních údajů člena, síť, platby, tisk, uživatelské relace a přiřazení kelímků. Cílem rozdělení do modulů je zvýšení přehlednosti.

Každý model bude dále rozdělen na domény, které popisují entity daného modulu. Repositáře slouží pro vyhledávání a ukládání entity v databázi. Servisní vrstva, též zvaná business vrstva, se stará o logiku aplikace. Ovladače, anglicky pojmenované Controllers, poskytují API pro komunikaci s klientskou částí aplikace. DTO je zkratkou pro Data Transfer Object. DTO pomáhá ovladačům převádět objekty na data a data na objekty.

Každá entita má generativní id a velká část entit má atributy createdAt, createdBy, createdComment, deletedAt, deletedBy, deletedComment a member, je tedy vhodné vytvořit abstraktní třídy, aby se předešlo duplikaci kódu, nejen v doménové vrstvě, ale i v navazujících vrstvách. Abstraktní doménové třídy jsou zobrazeny diagramem tříd 4.1.



Obrázek 4.1 Návrhový model tříd

4.3 Doménový model

Doménový model je strukturovaný popis klíčových konceptů, entit a vztahů v dané doméně, který slouží k porozumění a dokumentaci oblasti problému nebo řešení. Popisuje základní entity, které se vyskytují v systému a vztahy mezi nimi. Doménový model nového systému je inspirovaný doménovým modelem současného řešení a zohledňuje nové či pozměněné funkční požadavky. Schéma doménového modelu z důvodu přehlednosti nespojuje atributy createdBy a deletedBy s entitou Member.

Doménový modul je zobrazen pomocí kreslítka, které lze nalézt na adrese https://dbs.fit.cvut-.cz/kreslitko. Realizace pomocí kreslítka byla snadná, bohužel kombinace velkého počtu entit, neúspornost velikosti zobrazení entity a malá písmena, si vyžaduje diagram rozdělit. Doménový model je rozdělen podle navržených modulů. Diagramy modulů lze nalézt v přílohách:

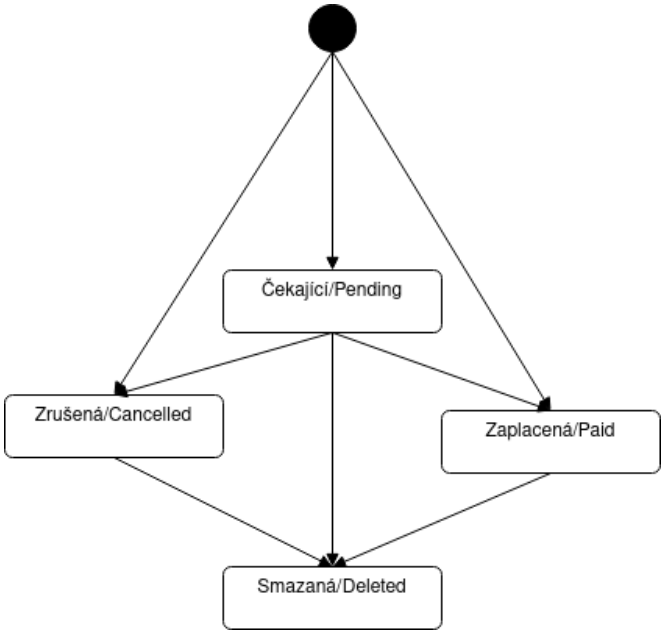
1. Doménový model platby A.3
2. Doménový model tisku A.4
3. Doménový model člena A.5
4. Doménový model kelímku, emailového aliasu a posilovny A.6
5. Doménový model sítě a uživatelské relace A.7
6. Doménový model voleb A.8

4.4 Stavový diagram

Třídy obsahující atributy, které souvisí se zpětně vyhledatelným smazáním (deletedAt, deletedBy, deletedComment) mají pouze dva vnitřní stavy, buďto je entita vytvořená a pak může být následně smazaná.

Nový systém zjednodušuje stavy pro platbu kartou, ve starém systému bylo několik stavů, které byly redundantní jako například ok a paid. Redundance vznikla integrací více platebních systémů. Nové stavy jsou čekající/pending, zrušená/cancelled, zaplacená/paid a smazaná/deleted, zobrazeny ve stavovém diagramu 4.2.

První tři reflektují stav platby dle platební brány. Poslední stav deleted může vytvořit pokladník. Stav označuje, jestli je platba smazaná v rámci našeho systému, například z důvodu chybného vytvoření. Je nutné zmínit, že pouze platba ve stavu zaplacená/paid vytváří členství a nabíjí konto. Pokud je platba v jiném stavu, mapování platby se neprovádí. V případě přechodu ze stavu zaplacená do stavu smazaná je nutné mapování platby smazat.



Obrázek 4.2 Stavový diagram

Kapitola 5

Implementace kolejního systému

Implementace je klíčovou fází v každém projektu v oblasti informatiky, jedná se o jedinou část vývoje softwarového produktu, kterou nelze přeskočit. Implementace představuje přechod od teoretického konceptu k praktickému provedení, kde se teorie mění ve funkční a použitelný výsledek.

5.1 Databáze

Struktura databáze je vytvářena v doménových složkách, kde jsou definované jednotlivé entity, které se namapují na tabulky. V souboru DataLoader.java je definováno naplnění databáze základními daty.

5.1.1 Uložení MAC adresy

Při implementaci se objevil problém s datovým typem macaddr, tento datový typ je určen k ukládání MAC adres. Datový typ macaddr používá PostrgreSQL, bohužel neexistuje jednoduchý způsob pro mapování textového řetězce na datový typ macaddr. Vzhledem k tomu, že macaddr zajišťuje především formát dat, tak bylo rozhodnuto tento datový typ opustit a implementovat vlastní omezení pro sloupec ukládající mac adresy. Řešení je zobrazeno v ukázce kódu 5.1.

```
@Column(name = "mac_address", nullable = false, length = 17)
@Pattern(regexp = "^([0-9a-f]{2}[:]){5}([0-9a-f]{2})$",
    message = "Invalid MAC address format")
private String macAddress;
```

Výpis kódu 5.1 Ukázka kódu ze souboru src/main/java/cz/cvut/fit/kvasvojt/sinis/modules/network/domain/Device.java

5.1.2 Zobrazení všech validních zařízení

Nejsložitější dotaz do databáze vyžaduje endpoint „GET /devices?isValid=true”, který vypisuje všechna validní síťová zařízení. Validní zařízení je vlastněné členem, který je držitelem platného

základního členství. Členství mohl získat odpuštěním platby nebo si ho koupil, samozřejmě, záznam platby ani informace o odpuštění platby nesmí být označeny jako smazané. Zároveň zařízení by nemělo být smazané ani zablokované. V neposlední řadě člen nesměl požádat o smazání účtu a správce sítě mu nezablokoval přístup k síti. Nelze opomenout internet zdarma na pár dní pro členy, kteří provedli před-registrace, nebo kterým byla ověřena identita. Dotaz si lze prohlédnout v ukázce kódu A.2.

5.2 Bezpečnost

Autorizaci nelze implementovat bez autentizace. Autentizace je ověření identity uživatele a autorizace je přidělení práv. Tento systém využívá pro autentizaci JWT.

5.2.1 JSON Web Token

JSON Web Token se často zkracuje na JWT [33]. Jedná se o otevřený standard RFC 7519, který definuje kompaktní a samostatný způsob bezpečného přenosu informací mezi stranami ve formě JSON objektu. Tato informace může být ověřena, protože je digitálně podepsaná. JWT může být podepsán pomocí tajemství například algoritmem HMAC nebo páru veřejného/soukromého klíče realizovaného RSA nebo ECDSA.

JWT mohou být šifrovány pro zajištění tajnosti mezi stranami. Podepsané tokeny mohou ověřit integritu tvrzení, zatímco zašifrované tokeny skrývají tato tvrzení před ostatními stranami. Když jsou tokeny podepsány pomocí páru veřejného/soukromého klíče, tak pouze strana, která drží soukromý klíč, je ta, která ho mohla podepsala. [33]

Spring Security poskytuje robustní podporu pro autentizaci pomocí JWT. Pomocí Spring Security lze snadno implementovat autentizaci a autorizaci založenou na JWT ve Spring Boot aplikaci. [34]

5.2.2 Autorizace

Autorizace je zajištěna anotací @PreAuthorize, tato anotace je vždy umístěna u endpointu, implementaci si lze prohlédnout v ukázce kódu 5.2. Většinou anotace volá funkci definovanou v souboru security/MyAuthorization v případě ukázkového příkladu je volána metoda ze třídy MemberAuthorization, která dědí z MyAuthorization. Tuto anotaci lze nalézt u každého endpointu, který není volně dostupný. Volně dostupné endpointy slouží k poskytnutí informací pro vyplnění před registračního formuláře, jako jsou například čísla pokojů nebo fakulty univerzit. Ukázka implementace autorizační funkce zobrazuje výpis kódu 5.3.

```
@PostMapping("/block-accesses")
@PreAuthorize("@memberAuthorization.canBlockAccess
        (#loginMember, #blockAccessCreateDto)")
public ResponseEntity<BlockAccessResponseDto> create(
        @Validated @RequestBody BlockAccessCreateDto blockAccessCreateDto,
        @AuthenticationPrincipal Member loginMember
)
```

Výpis kódu 5.2 Ukázka kódu ze souboru src/main/java/cz/cvut/fit/kvasvojt/sinis/modules/member/controller/BlockAccessController.java

```
public boolean canBlockAccess(
        Member loginMember,
        BlockAccessCreateDto blockAccessCreateDto
) {

    if (loginMember.hasRole(RoleTypeEnum.ROLE_NET_ADMIN) &&
            blockAccessCreateDto.getBlockAccessType() ==
            BlockAccessTypeEnum.NETWORK_ACCESS) {
        return true;
    }

    if (loginMember.hasRole(RoleTypeEnum.ROLE_GYM_ADMIN) &&
            blockAccessCreateDto.getBlockAccessType() ==
            BlockAccessTypeEnum.GYM_ACCESS) {
        return true;
    }
    return false;
}
```

Výpis kódu 5.3 Ukázka kódu ze souboru src/main/java/cz/cvut/fit/kvasvojt/sinis/modules/member/security/MemberAuthorization.java

5.3 Rozdělení kódu

Systém je strukturován do modulů. Už funkční požadavky jsou rozděleny dle modulů. Každý modul je dělen na repositáře, doménu, servisy, bezpečnost, kontrolory a dto. Domény definují entity, které se pomocí JPA [29] přemění na tabulky v databázi. Spousta entit je vytvořena administrátorem nebo členem a je vhodné uchovávat informaci, o tom kdo danou entitu vytvořil, proto je definována abstraktní třída RecordableEntity.

Většina entit, která je vytvořena administrátorem, může být i smazána, avšak i smazané entity musí být uloženy pro zpětné vyhledání. Příkladem je nabití konta pomocí hotovosti a následné smazání. Člen by tvrdil, že hotovost zaplatil a administrátor by si stál za tím, že ho nikdy neviděl a peníze by si nechal pro sebe. Tuto funkcionalitu zajišťuje třída RecordableDeleteEntity, která dědí z RecordableEntity.

Člen má relaci s nemalým počtem entit, a proto se vyplatí vytvořit abstraktní třídu RecordableMember. Pro většinu entity platí, že pokud mají vztah se členem, tak je tato entita vytvářena a mazána členem nebo administrátorem, proto RecordableMember dědí z RecordableDeleteEntity.

V repositářích jsou definovány dotazy do databáze. Pro třídy, které dědí z RecordableDeleteEntity existuje repositář se jménem RecordableDeleteEntityJpaRepository, v němž jsou v definovány metody pro výpis všech smazaných a nesmazaných záznamů této entity. Třídy, které dědí z RecordableMember mají připravený repositář MemberRecordJpaRepository. Tento repositář umožňuje filtrovat záznamy podle identifikátoru člena a podle toho, jestli není smazán.

Servisy jsou definované ve složce business. Na abstraktní repositáře navazují abstraktní servisy, ale i ty dědí ze třídy AbstractService, která implementuje základní metody, jako například create, update a getAll. Servisy obsahují logiku aplikace jako třeba mapování plateb. Za zmínku stojí smazání namapované platby. Pokud se smaže namapovaná platba je třeba smazat i mapování a případně označit za smazanou i transakci s kredity. Kredity slouží pro tisknutí dokumentů na tiskárně.

Kontrolory definují API. Kontrolory jsou závislé na servisní vrstvě aplikace. Na abstraktní

servisy navazují abstraktní kontrolory. Třída AbstractDeleteController implementuje metodu deletion, jejíž funkcí je označit entitu za smazanou. V tomto systému se kontrolory starají o mapování objektů na data a data na objekty. Pro mapování je využit ModelMapper [35] a DTO objekty definované ve stejnojmenné složce.

Autorizace se nalézá ve složce security. V současné době má tato složka pouze jednu třídu, jež dědí z MyAuthorization. Některé moduly ani tuto složku nemají, protože veškeré autorizační funkce, které modul vyžaduje jsou už definovány v MyAuthorization. Jak už se dá z předešlého textu odvodit, účelem bezpečnosti je povolovat a zakazovat členům a administrátorům přístup ke koncovým bodům aplikace.

Kapitola 6
Testování

Obvykle po implementaci následuje testování. Tato kapitola se zabývá testováním implementované serverové části aplikace. Nejdříve se vygenerovala API dokumentace, která se nahrála do aplikace Postman a následně se vytvořily testy pro pozitivní průchod.

6.1 Automatizované testování

Pomocí aplikace Postman se kontrolují pozitivní odpovědi na předdefinované REST API žádosti. U odpovědi se v testech tohoto systému kontroluje status a tělo HTTP odpovědi.

6.1.1 Popis skriptů

Postman umožňuje definování několik různých skriptů. Pre-request skript je vykonán před odesláním požadavku. Post-response skript je vykonán po provedení požadavku a přijmutí odpovědi. Pre-request skript a post-response skript lze definovat na úrovni celé kolekce, v tom případě se skripty provedou před a po každém požadavku. Dají se definovat skripty na úrovni složky, v tomto projektu jsou endpointy shromažďované do složek podle tagu a tagy representují moduly. Lze tedy definovat pre-request a post-response na každý požadavek v rámci jednoho modulu. Samozřejmě Postamn umožňuje definovat pre-request a post-response skript na jeden jediný požadavek. [36]

Pre-request skript připravuje systém do optimálního stavu, například před vydáním kelímku, je pomocí něj členovi zakoupeno členství. Post-response skript se následně využije pro otestování odpovědi.

6.1.2 Popis proměnných

Proměnné umožňují ukládat a znovu používat hodnoty v Postmanu. Uložením hodnoty jako proměnné umožňuje na ni odkazovat v kolekcích, prostředích, požadavcích a testovacích skriptech. Proměnné pomáhají pracovat efektivně, spolupracovat s kolegy a nastavit dynamické pracovní postupy. [37]

6.1.3 Viditelnost proměnných

1. Globální proměnné umožňují přistupovat k datům mezi sbírkami, požadavky, testovacími skripty a prostředími.

2. Proměnné sbírky jsou k dispozici pro každý požadavek ve sbírce a jsou nezávislé na prostředích. Proměnné sbírky se nemění na základě vybraného prostředí.

3. Proměnné prostředí umožňují rozdělit proměnné do různých prostředí, například lokální vývoj versus testování nebo produkce.

4. Proměnné dat pocházejí z externích souborů CSV a JSON k definici sad dat. Proměnné dat mají aktuální hodnoty, které přetrvávají požadavek nebo běh sbírky.

5. Lokální proměnné jsou dočasné proměnné, ke kterým se přistupuje ve skriptech požadavků. Hodnoty lokálních proměnných jsou omezeny na jediný běh požadavku nebo sbírky a po skončení běhu již nejsou k dispozici.

Pokud je proměnná se stejným názvem deklarována vícekrát s různou viditelností, bude použita hodnota uložená v proměnné s nejúžeji definovanou viditelností. Například, pokud existuje globální proměnná s názvem username a lokální proměnná s názvem username, při spuštění požadavku bude použita hodnota lokální proměnné. V testování kolejního systému jsou využity lokální proměnné a proměnné definované v rámci sbírky. [37]

6.1.4 Testování statusu odpovědi

Testování statusu je implementováno v testovacím skriptu celé kolekce. Testování statusu je vcelku jednoduché, protože všechny požadavky obsahující metodu POST vrací 201 s výjimkou endpointu login, který vrací 200. Požadavky provedené metodou GET, PUT a PATCH vrací status 200. Implementaci testu statusů si lze prohlédnout v ukázce kódu 6.1.

```
if (
    pm.request.method === "GET" ||
    pm.request.method === "PATCH" ||
    pm.request.method === "PUT" ||
    pm.request.url.path[0] == "login"
) {
    pm.test("Status code is 200", function () {
        pm.response.to.have.status(200);
    });
} else if (pm.request.method === "POST") {
    pm.test("Status code is 201", function () {
        pm.response.to.have.status(201);
    });
}
```

Výpis kódu 6.1 Ukázka testování statusu odpovědi za pomoci aplikace Postman

6.1.5 Testování těla odpovědi

Spousta odpovědí má podobné atributy, jako je například atribut createdAt. Hodnota tohoto atributu v případě požadavku vytvoření nového zdroje musí mít časovou hodnotu, která bude rovna nebo starší oproti současnosti a zároveň nesmí být příliš stará. V případě tohoto požadavku je vhodné neopakovat kód a test implementovat ve skriptu společném pro všechny požadavky. Existují však i unikátní požadavky.

Pro ukázku testu, který je zaměřen na obsah těla odpovědi, je vybrán koncový bod „GET /membership-types", který vypíše typy členství. Tento endpoint bude mít vždy stejnou odpověď s téměř konstantní hodnotou atributů. Cena či popis členství se rozhodně nebudou měnit každý semestr. Proto se kontroluje celé tělo odpovědi, tento test zobrazuje výpis kódu 6.2.

```
let expectedResponseBody =
    [
        {
            "membershipType": "BASIC_MEMBERSHIP",
            "description": "Členství v klubu Sincoolka. " +
                "Možnost volit a být volen do Představenstva. " +
                "Možnost půjčovat si majetek klubu k tomu určený. " +
                "Připojení do akademické sítě. " +
                "Možnost využití kolejní tiskárny.",
            "price": 1000
        },
        {
            "membershipType": "SPORTS_MEMBERSHIP",
            "description": "Přístup do posilovny a do sauny. " +
                "Vyžaduje platné Základní členství.",
            "price": 500
        }
    ]


pm.test("Response body contains expected data", function () {
    response = pm.response.json();
    pm.expect(response).to.deep.equal(expectedResponseBody);
});
```

Výpis kódu 6.2 Ukázka testování těla odpovědi za pomoci aplikace Postman

6.2 Manuální testování

Manuální testování probíhalo při vývoji aplikace. Všechny negativní scénáře byly znovu otestovány na konci vývoje. Negativním scénářem je myšleno, že odpověď na požadavek nevrátí 200 nebo 201, ale například 404, pokud adresa není nalezena, většinou se jedná o nenalezení entity s id, o které je požádáno v dotazu. Konflikt je ohlášen statusem 409, tuto odpověď lze očekávat při pokusu označit za smazanou již smazanou entitu, nebo při přidělení kelímku členovi, který už vlastní kelímek a mnoho dalších konfliktních dotazů. Na špatnou žádost se vrací kód 400, příkladem jest absence nutného parametru. Dále je manuálně otestované párování plateb a jejich mazání. Následovalo testování autorizace a autentizace žádostí.

6.3 Shrnutí testování

Automatické testy API pomocí aplikace Postman umožňují být spouštěny opakovaně s minimálními náklady. V současném stavu všechny testy prochází. Testy jsou implementované tak, že i po opakovaném spuštění nenastane selhání, příkladem může být vytvoření semestru, kde atributy pro mapování platby nesmí vytvořit časový konflikt, neboli v jakémkoliv časovém okamžiku se platba namapuje maximálně na jeden semestr. Pomocí manuálních testů jsou otestovány nesprávné žádosti. Lze tedy aplikaci považovat za funkční, ale je nutné zmínit, že testy nikdy nemohou prokázat bezchybovost aplikace, pouze mohou odhalit přítomnost chyby.

Kapitola 7

Dokumentace a další vývoj

7.1 Dokumentace

Dokumentace je generována za pomoci knihovny spring-doc popsané v sekci 4.1.7. Tato knihovna se sama snaží vygenerovat dokumentaci, ale občas potřebuje napovědět anotacemi. V následující ukázce 7.1 se pomocí anotace @Operation nastaví souhrn a popis jednoho endpointu, nelze opomenout to, že knihovna spring-doc je schopná využít anotaci @ResponseStatus(HttpStatus.OK) a s její pomocí vygeneruje odpověď na požadavek se statusem 200. Knihovna automaticky přidá k odpovědi se statusem 200 tělo, které bude respektovat strukturu třídy RoomResponseDto. Ve třídě RoomResponseDto jsou definované anotace @Schema(example = ”99”), která do odpovědi přidají ukázkové hodnoty.

Dokumentaci ve formátu YAML poskytuje endpoint „GET /v3/api-docs.yaml”. Zobrazení dokumentace lze zakázat pomocí příkazu springdoc.api-docs.enabled=false v souboru application.properties.

```
@Operation(
    summary = "Get all rooms",
    description = "This endpoint return all rooms, which can be be possibly
        link to member."
)
@ResponseStatus(HttpStatus.OK)
@GetMapping("/rooms")
public Collection<RoomResponseDto> getAll() {
    return convertManyToDTO(roomService.getAll());
}
```

Výpis kódu 7.1 Ukázka kódu ze souboru src/main/java/cz/cvut/fit/kvasvojt/sinis/modules/network/controller/RoomController.java

7.2 Další vývoj

Pro nasazení nového kolejního systému, je nutné, aby někdo navázal na tuto bakalářskou práci a navrhl a implementoval klientskou část aplikace. Její návrh a implementace však nebude velkým problémem, protože tato práce definuje a testuje REST API, pomocí něhož lze dopsat zbylou část. Při návrhu a grafického rozhraní se lze inspirovat současným řešením kolejního klubu. Webové stránky klubu sice nejsou vůbec responsivní, ale základní struktura je více než inspirativní. Na současné klientské části lze především ocenit QR kód pro platbu členských příspěvků, kterou by členové rozhodně uvítali i v budoucí klientské části.

Potřeba jsou dodělat nebo přepsat existující skripty, která integrují tento systém s již zavedenými strukturami. Příkladem takového skriptu může být stahování plateb ze serveru banky, nebo skript pro komunikaci se čtečkou. Další změnou, kterou bude nutno provést je autorizace uživatele při používání sítě WIFI, nový systém totiž nepodporuje MD4 hash, tento fakt byl samozřejmě konzultován už při vytváření analýzy. Tuto změnu však lze považovat za jednoznačné bezpečnostní zlepšení, protože se uživatelské heslo nebude nikde ukládat jako MD4 hash, která je z dnešního pohledu nedostačující.

Kolejní klub zvažuje rezervační systém pro multimediální místnost. Pravděpodobně bude nasazené nějaké open-source řešení. Pokud se tak stane, bylo by vhodné rezervační systém propojit se systémem implementovaném v této bakalářské práci.

Závěr

Hlavním cílem této práce bylo zanalyzovat, navrhnout a implementovat serverovou část informačního systému pro kolejní klub Sincoolka. Tento nový systém má nahradit stávající. V této bakalářské práci je tedy úspěšně implementován systém, který svojí funkčností pokrývá stávající řešení kolejního klubu.

Tato práce začala popisem softwarového řešení současného systému. Systém je vyvíjen dlouhou dobu a má spoustu nedostatků, kterým je nutno se vyvarovat. Při seznámení se softwarovým řešením, lze lehce nahlédnout do funkčních požadavků systému, čehož lze využít pro analýzu nového systému. Dále byly popsány technologie pomocí nichž byl současný systém vyvíjen. Následoval popis databázové schématu, základní rozdělení systému a závažné nedostatky. Při popisu bylo prioritizováno zahrnutí klíčových aspektů, aby práce zůstala srozumitelná a rozumně dlouhá pro čtenáře.

Následovala analytická část. Součástí analýzy bylo krátké nahlédnutí do podobných systémů, jednalo se o systémy provozované na kolejních klubech ČVUT. U funkčních požadavků je přidán atribut, zdali se tato funkčnost objevuje i v aktuálně nasazeném systému, či se bude jednat o novou funkcionalitu. Po funkčních požadavcích následovaly případy užití a diagram aktivit.

Po analýze přichází na řadu návrh, ve kterém jsou popsány použité technologie a důvody pro jejich vybrání a použití. Následuje popis struktury systému a menší diagram zobrazující abstraktní třídy určené pro doménovou vrstvu. Nakonec je popsán stavový diagram pro platbu kartou.

Za návrhem následuje implementace. V implementaci je popsán problém s datovým typem, který se objevil při implementaci modelové třídy Device, a jeho řešení. Dále je v implementaci popsán nejsložitější dotaz, který se v projektu vyskytuje. V sekci bezpečnost je popsána autentizace a autorizace. V neposlední řadě je popsáno rozdělení kódu, přičemž je kladen důraz zejména na abstraktní třídy.

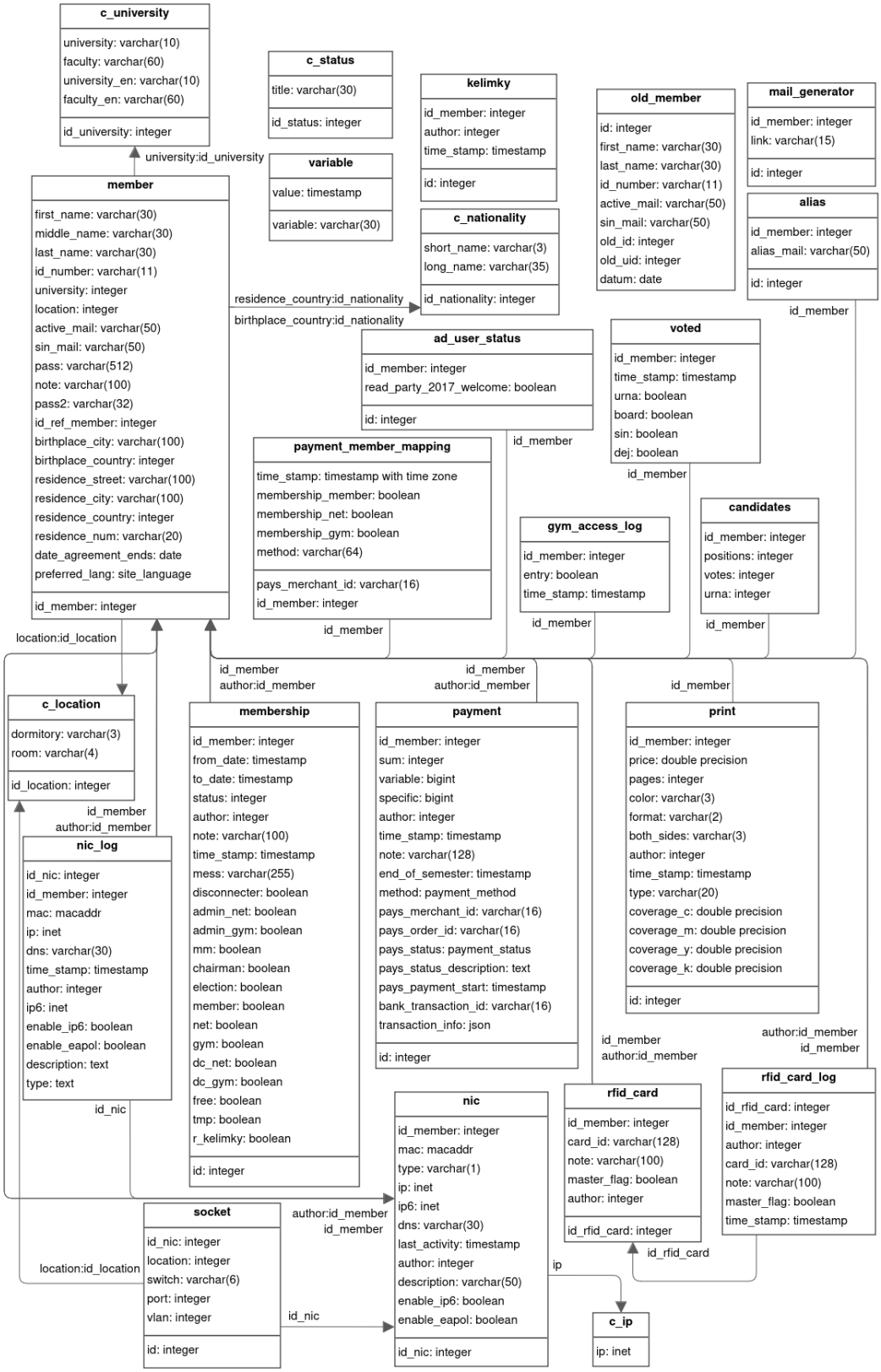
Testy úspěšně ověřily funkčnost implementace. REST API je testováno pomocí aplikace Postman. Postman využívá JavaScript k testování positivního průchod aplikací, negativní odpovědi jsou otestovány ručně.

Dokumentace je generována přímo z kódu za pomoci knihovny spring-doc. Vygenerovanou dokumentaci lze například nahrát do Postmanu a následně intuitivně posílat dotazy na API.

Všechny cíle a požadavky byli splněny. Systém nad rámec cílů implementuje funkční požadavky, které se nevyskytují v současném systému, především se jedná o uživatelské relace a organizaci voleb. K nasazení aplikace ještě chybí klientská část, která však nebyla součástí této práce. Nové uživatelské rozhraní může být inspirováno současnou webovou stránkou. Díky otestovanému a zdokumentovanému API nebude dodělání klientské části představovat výrazně náročnou záležitost.

Příloha A

Přílohy

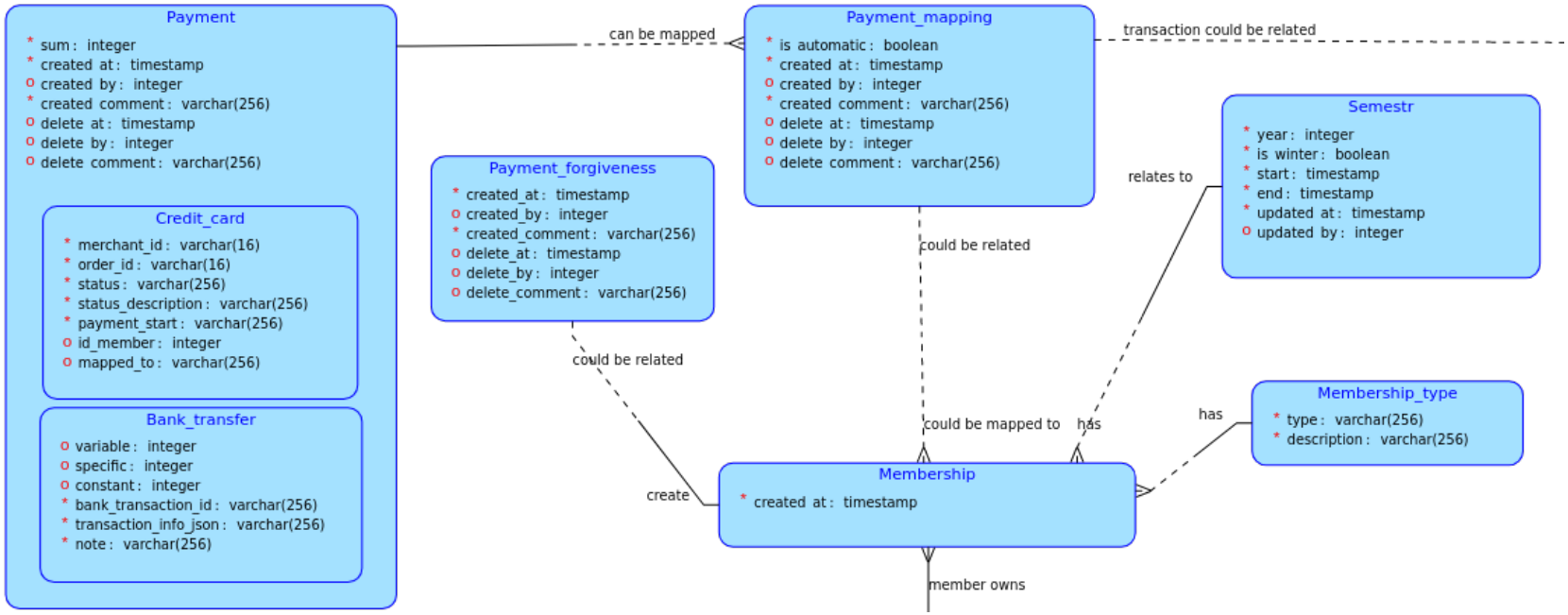


Obrázek A.1 Databázové schéma současně používaného systému

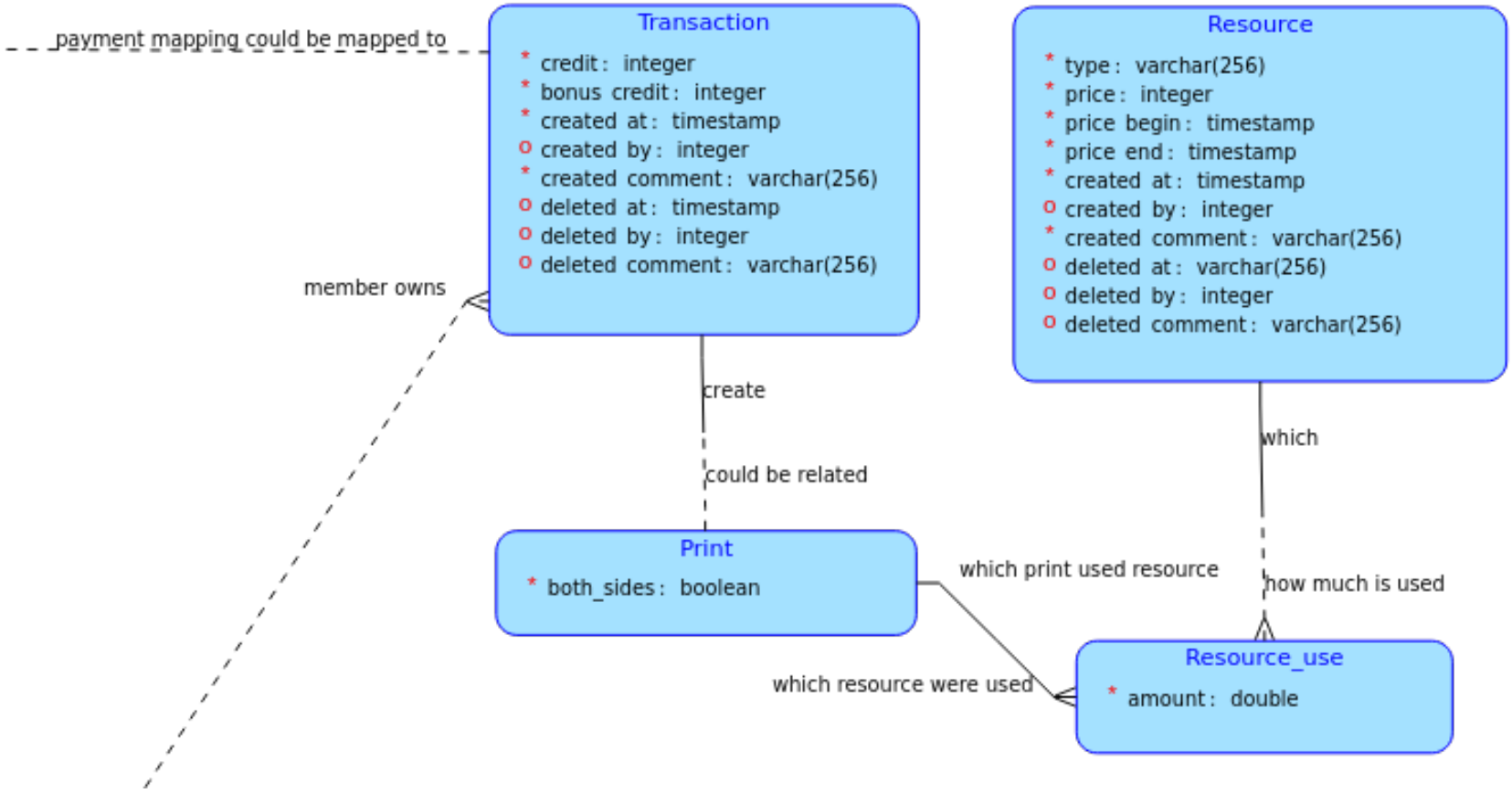
```php
<?php
$SQLnew->beginTransaction();
try {
    //kontrola, jestli neni sin_mail uz pouzit
    $result = $SQLnew->query("SELECT * FROM member
                              WHERE sin_mail = ?", $sin_mail);
    $mail_exists = $result->getRowCount() > 0;
    $posuvnik = 1;
    while($mail_exists) {
        $sin_name = $first_name . "." . $last_name . $posuvnik;
        $sin_name = str_replace(" ", "", $sin_name);
        $sin_mail = $sin_name . "@sin.cvut.cz";
        $sin_mail = str_replace(" ", "", $sin_mail);

        //iterujeme pres maily, az narazime na takovy, ktery neexistuje
        $result = $SQLnew->query("SELECT id_member FROM
                                  member WHERE sin_mail = ?", $sin_mail);
        $mail_exists = $result->getRowCount() > 0;
        $posuvnik++;
    }
    //jednotny format datumu
    $_SESSION["reg_data"]["id_number"] =
            transformDATE($_SESSION["reg_data"]["id_number"]);
    //vlozeni uzivatele do DB
    $SQLnew->query("INSERT INTO member", array(
        'first_name' => $_SESSION["reg_data"]["first_name"],
        ...
        'active_mail' => $_SESSION["reg_data"]["active_mail"],
        'sin_mail' => $sin_mail,
        'pass2' => 'temp'
        ));
    //zjisteni nainkrementovaneho ID_member
    $id_member = $SQLnew->getInsertId();
    //nastaveni jazyka
    $lang = $_SESSION["lang"] = $_SESSION["reg_data"]["preferred_lang"];
$SQLnew->commit();
} catch(Exception $e) {
    $SQLnew->rollback();
    //vysvetlujici chybova hlaska a kontakty
    error_500($e);
}
//pridani hesla
$SQLnew->query("UPDATE member SET pass2 = ? WHERE id_member = ?",
    NTLMHash($_SESSION["reg_data"]["passwd"]),
    $id_member
);
?>
```

Výpis kódu A.1 Ukázka kódu ze souboru usr_iface/volby.php

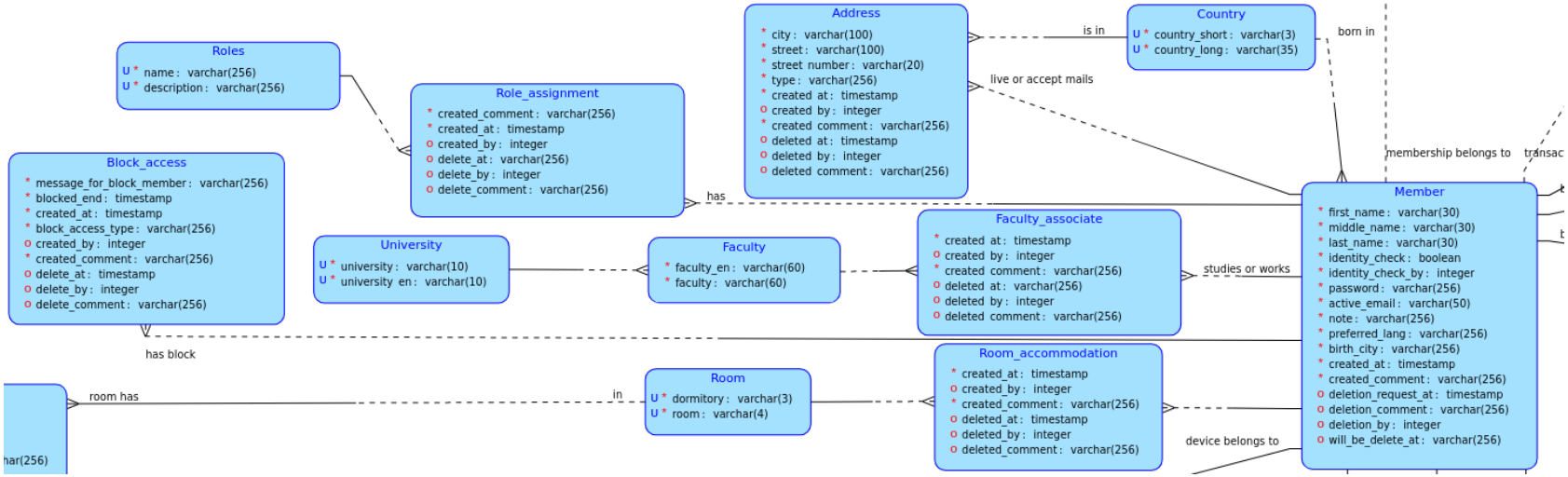
Obrázek A.2 Doménový model informačního systému využívaného na Masařce



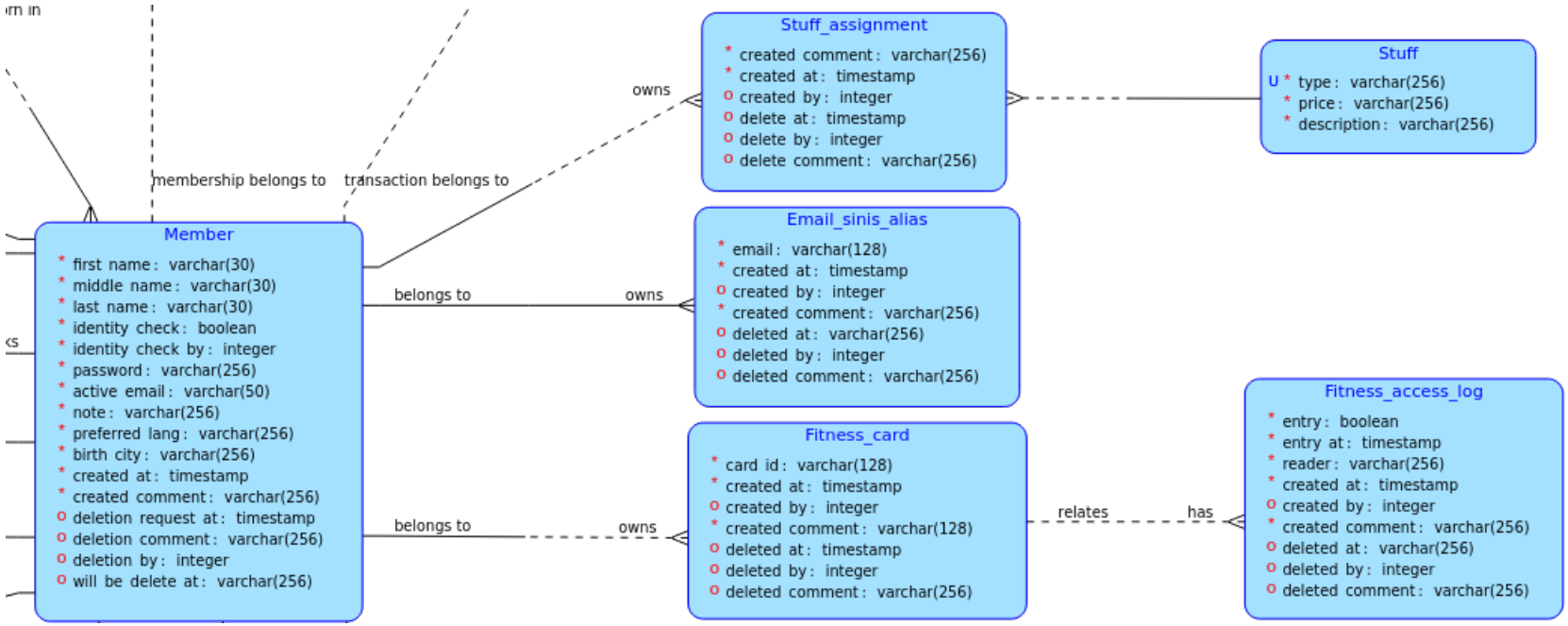
Obrázek A.3 Doménový model platby



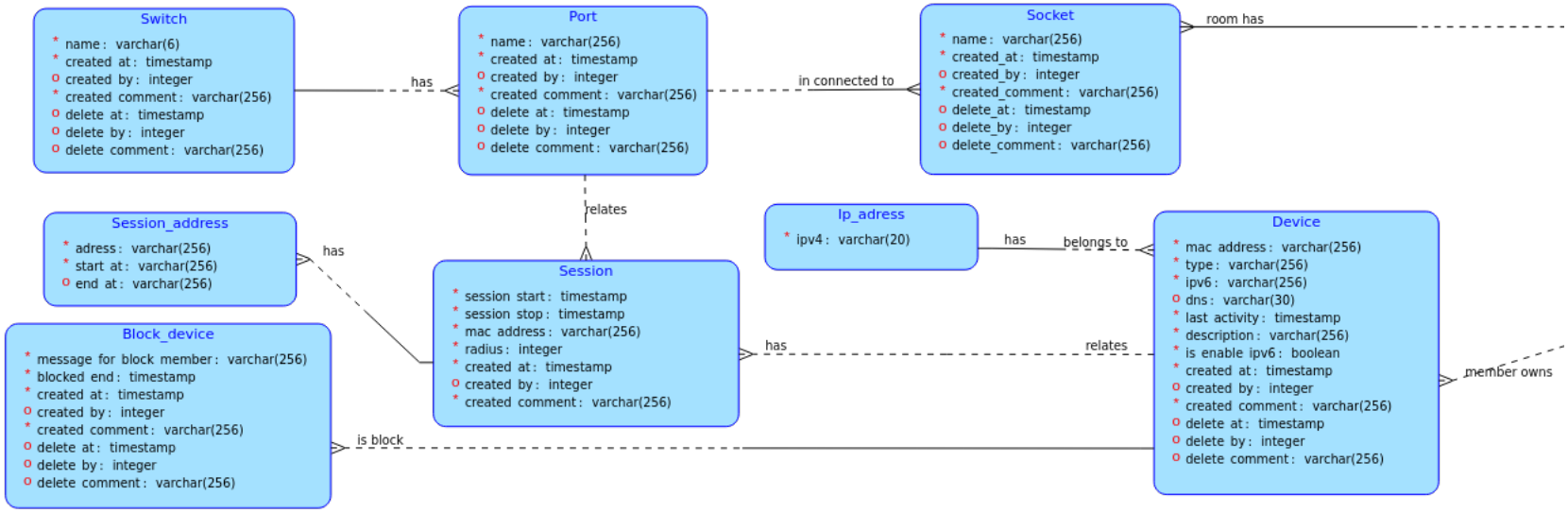
Obrázek A.4 Doménový model tisku



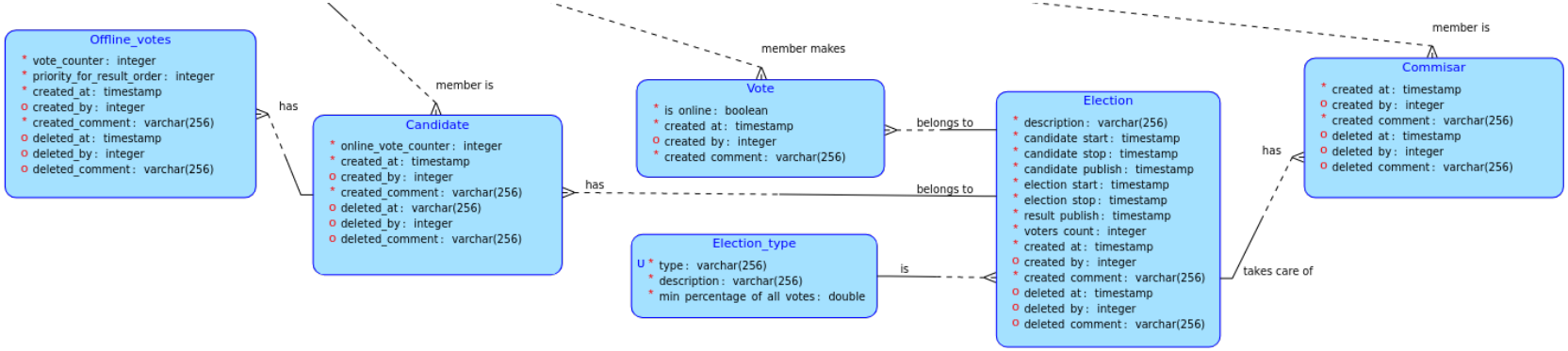
Obrázek A.5 Doménový model člena



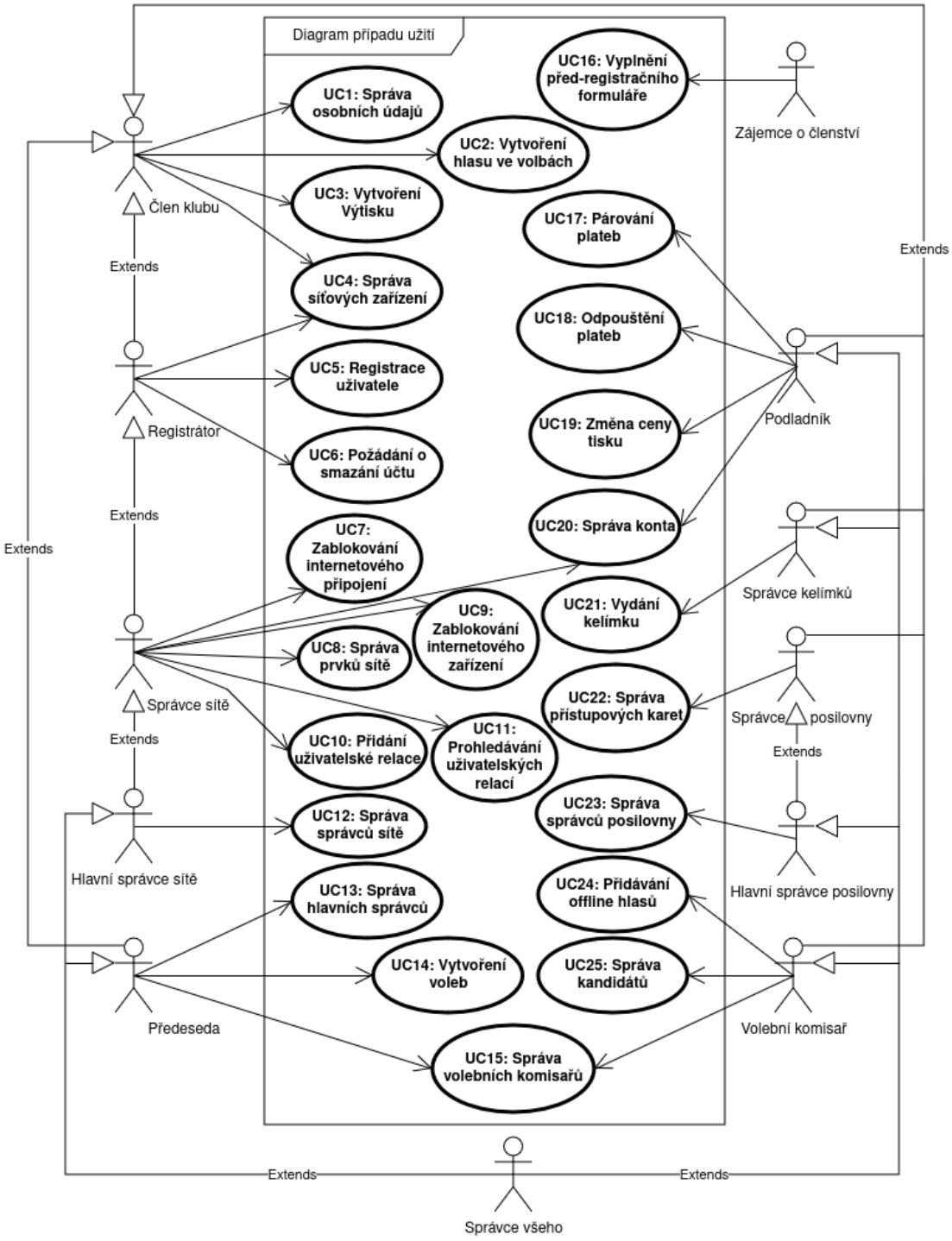
Obrázek A.6 Doménový model kelímku, emailového aliasu a posilovny



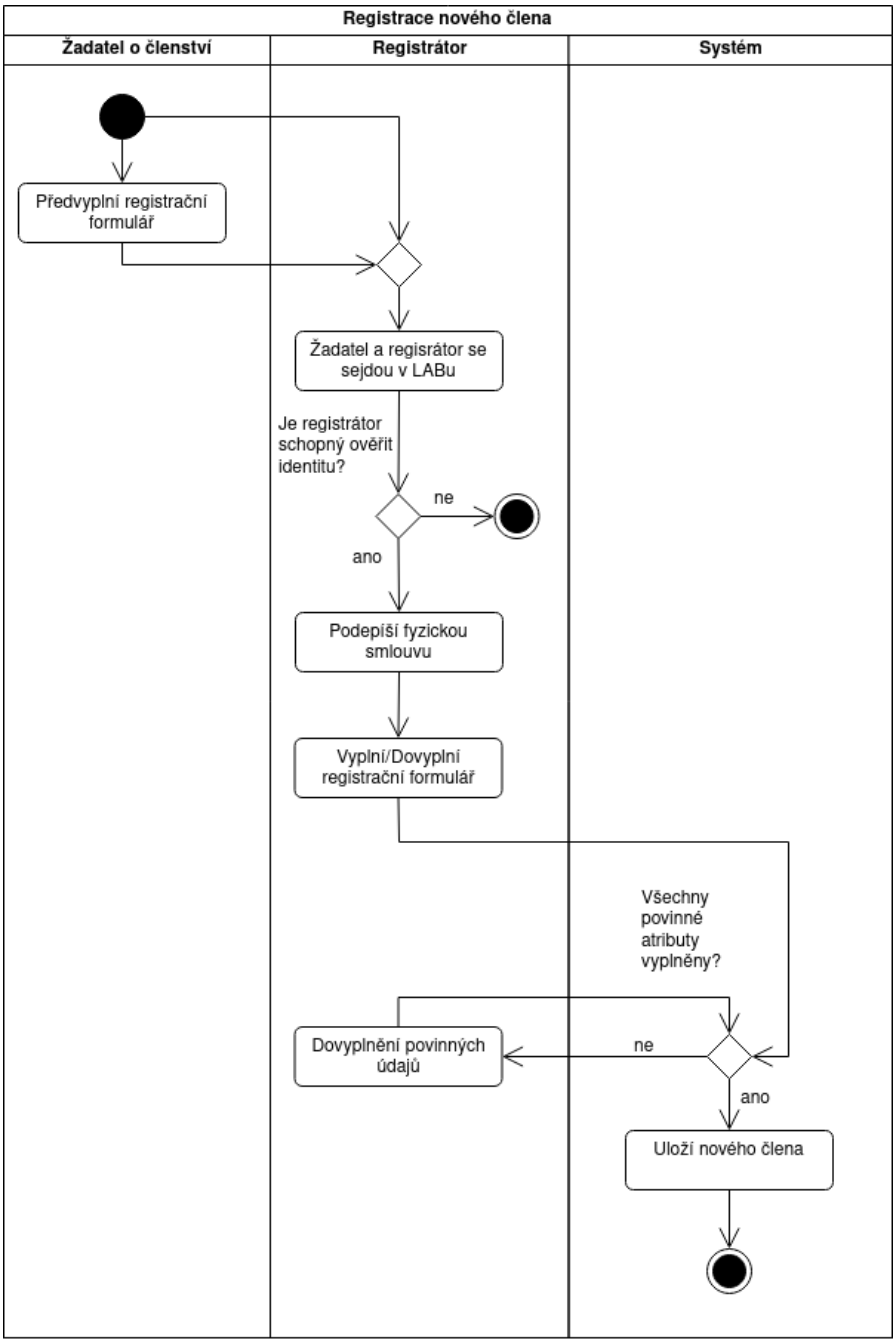
Obrázek A.7 Doménový model sítě a uživatelské relace



Obrázek A.8 Doménový model voleb



Obrázek A.9 Případy užití



Obrázek A.10 Aktivity diagram registrace člena

```
@Query(
"SELECT d FROM Device d " +
"JOIN BlockDevice bd ON bd.device = d " +
"LEFT JOIN Membership ms ON d.member = ms.member " +
"JOIN BlockAccess ba ON ba.member = d.member " +
"JOIN Member member ON d.member = member " +
"JOIN MembershipType mt ON ms.membershipType = mt " +
"LEFT JOIN PaymentMembershipForgiveness pf ON pf.membership = ms " +
"LEFT JOIN PaymentMapping pm ON pm = ms.paymentMapping " +

// free internet
"WHERE " +
"((( member.identityCheck = true AND " +
" (DATEDIFF(day, member.identityCheckAt, CURRENT_DATE())" +
        " < :freeIdentityCheck)) " +
" OR (DATEDIFF(day, member.createdAt, CURRENT_DATE())" +
        " < :freePreRegistration)) " +

// has valid membership and payment is not deleted
" OR ( " +
"  ms.semester.startOfMappingPayment < CURRENT_TIMESTAMP " +
"  AND CURRENT_TIMESTAMP < ms.semester.endOfSemester " +
"  AND ms.membershipType.membershipType = :membershipType " +
"  AND " +
"   (( " +
"    pm IS  NULL " +
"    AND pf IS NOT NULL " +
"    AND pf.deletedAt IS NULL " +
"    ) OR (" +
"    pf IS  NULL " +
"    AND pm  IS NOT NULL " +
"    AND pm.deletedAt IS NULL " +
"))))" +

// is access not blocked
" AND NOT( ba.blockAccessType = :blockAccessTypeEnum " +
" AND CURRENT_TIMESTAMP < ba.blockedEnd AND ba.deletedAt IS NULL ) " +

// is device not blocked
"AND  NOT ( CURRENT_TIMESTAMP < bd.blockedEnd AND bd.deletedAt IS NULL ) " +
"AND d.deletedAt IS NULL "
)
Iterable<Device> findValid(
    @Param("membershipType") MembershipTypeEnum membershipType,
    @Param("blockAccessTypeEnum") BlockAccessTypeEnum BlockAccessTypeEnum,
    @Param("freePreRegistration") Integer
    freeInternetAccessAfterPreRegistration,
    @Param("freeIdentityCheck") Integer freeInternetAccessIdentityCheck
);
```

Výpis kódu A.2 Ukázka kódu ze souboru src/main/java/cz/cvut/fit/kvasvojt/sinis/modules/network/repository/DeviceRepository.java

Bibliografie

1. THE PHP GROUP. *What is PHP?* [Online]. 2021. [cit. 2024-03-29]. Dostupné z: `https://www.php.net/manual/en/intro-whatis.php`.
2. THE PHP GROUP. *What can PHP do?* [Online]. 2021. [cit. 2024-03-29]. Dostupné z: `https://www.php.net/manual/en/intro-whatcando.php`.
3. GRUDL, David. *Začínáme s Latte* [online]. 2008. [cit. 2024-03-30]. Dostupné z: `https://latte.nette.org/cs/`.
4. GRUDL, David. *7 důvodů, proč používat Nette* [online]. 2008. [cit. 2024-03-30]. Dostupné z: `https://nette.org/cs/10-reasons-why-nette`.
5. KIRSTENS A SPOL. *Cross Site Scripting (XSS)* [online]. 2024. [cit. 2024-04-02]. Dostupné z: `https://owasp.org/www-community/attacks/xss`.
6. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. *About* [online]. 1996. [cit. 2024-03-30]. Dostupné z: `https://www.postgresql.org/about/`.
7. INDIVIDUAL MOZILLA.ORG CONTRIBUTORS. *JavaScript* [online]. 1998. [cit. 2024-04-09]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/JavaScript`.
8. INDIVIDUAL MOZILLA.ORG CONTRIBUTORS. *First-class Function* [online]. 1998. [cit. 2024-04-09]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Glossary/First-class_Function`.
9. ORACLE CORPORATION. *Chapter 1. Introduction* [online]. [cit. 2024-04-18]. Dostupné z: `https://docs.oracle.com/javase/specs/jls/se9/html/jls-1.html`.
10. OPENJS FOUNDATION AND JQUERY CONTRIBUTORS. *What is jQuery?* [online]. 2024. [cit. 2024-03-30]. Dostupné z: `https://jquery.com/`.
11. SPRYMEDIA LTD. *Manual* [online]. 2007. [cit. 2024-03-30]. Dostupné z: `https://datatables.net/manual/`.
12. ATLASSIAN. *What is Git?* [online]. 2024. [cit. 2024-03-30]. Dostupné z: `https://www.atlassian.com/git/tutorials/what-is-git`.
13. *What is a database schema?* [online]. 2024. [cit. 2024-03-30]. Dostupné z: `https://www.ibm.com/topics/database-schema`.
14. LU CHEN A SPOL. *Základy normalizace databáze* [online]. 2023-07. [cit. 2024-03-30]. Dostupné z: `https://learn.microsoft.com/cs-cz/office/troubleshoot/access/database-normalization-description`.
15. DIONYSIA LEMONAKI. *Snake Case VS Camel Case VS Pascal Case VS Kebab Case – What's the Difference Between Casings?* [online]. 2022-11. [cit. 2024-04-21]. Dostupné z: `https://www.freecodecamp.org/news/snake-case-vs-camel-case-vs-pascal-case-vs-kebab-case-whats-the-difference/#snake-case`.

16. IVAN BELCIC. *What Is SQL Injection and How Does It Work?* [online]. 2024-10. [cit. 2024-03-30]. Dostupné z: https://www.avast.com/c-sql-injection.
17. CHEAT SHEETS SERIES TEAM. *Password Storage Cheat Sheet* [online]. 2024. [cit. 2024-03-30]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html.
18. BC. MARTIN DENDIS. *Registr členů klubu Hlávkova kolej* [online]. 2024. [cit. 2024-03-30]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/73949/F3-DP-2018-Dendis-Martin-registr_clenu_klubu_hlavkova_kolej.pdf?sequence=-1&isAllowed=y.
19. BROADCOM. *Why Spring?* [online]. 2005. [cit. 2024-04-05]. Dostupné z: https://spring.io/why-spring.
20. EVAN YOU. *The Progressive JavaScript Framework* [online]. 2024. [cit. 2024-04-29]. Dostupné z: https://vuejs.org.
21. *What is a REST API?* [online]. 2024. [cit. 2024-04-11]. Dostupné z: https://www.ibm.com/topics/rest-apis.
22. PRODUCTPLAN. *MoSCoW Prioritization* [online]. 2024. [cit. 2024-04-15]. Dostupné z: https://www.productplan.com/glossary/moscow-prioritization/.
23. RITWIK VERMA. *Kotlin vs Java – The Ideal Choice for Your Next Project in 2024* [online]. 2023-12. [cit. 2024-04-06]. Dostupné z: https://www.bacancytechnology.com/blog/kotlin-vs-java.
24. TIM LINDHOLM A SPOL. *Chapter 1. Introduction* [online]. 2015-02. [cit. 2024-04-06]. Dostupné z: https://docs.oracle.com/javase/specs/jvms/se8/html/jvms-1.html.
25. GRADLE INC. *Gradle Build Tool* [online]. 2023. [cit. 2024-04-08]. Dostupné z: https://docs.gradle.org/current/userguide/userguide.html.
26. HARESH AKALANKA. *Maven vs Gradle: A Comprehensive Comparison for Spring Boot Developers* [online]. 2023-02. [cit. 2024-04-08]. Dostupné z: https://akalankaih19.medium.com/maven-vs-gradle-a-comprehensive-comparison-for-spring-boot-developers-a04136562e.
27. DOCKER INC. *Docker Compose overview* [online]. 2013. [cit. 2024-04-08]. Dostupné z: https://docs.docker.com/compose.
28. DOCKER INC. *Why use Compose?* [online]. 2013. [cit. 2024-04-08]. Dostupné z: https://docs.docker.com/compose/intro/features-uses.
29. BROADCOM INC. *Spring Data JPA* [online]. 2005. [cit. 2024-04-08]. Dostupné z: https://spring.io/projects/spring-data-jpa.
30. *springdoc-openapi v2.5.0* [online]. 2024-03. [cit. 2024-04-20]. Dostupné z: https://springdoc.org/#Introduction.
31. POSTMAN, INC. *What is Postman?* [online]. 2011. [cit. 2024-04-09]. Dostupné z: https://www.postman.com/product/what-is-postman/.
32. LINKEDIN, INC. *What is Postman?* [online]. 2011. [cit. 2024-04-09]. Dostupné z: https://www.linkedin.com/advice/1/what-best-way-choose-programming-language-api.
33. *Introduction to JSON Web Tokens* [online]. 2024. [cit. 2024-04-08]. Dostupné z: https://jwt.io/introduction.
34. BROADCOM INC. *OAuth 2.0 Resource Server JWT* [online]. 2025. [cit. 2024-04-12]. Dostupné z: https://docs.spring.io/spring-security/reference/servlet/oauth2/resource-server/jwt.html.
35. JONATHAN HALTERMAN A SPOL. *modelmapper* [online]. 2011. [cit. 2024-04-09]. Dostupné z: https://modelmapper.org/.

36. POSTMAN, INC. *Scripting in Postman* [online]. 2024. [cit. 2024-04-24]. Dostupné z: https://learning.postman.com/docs/writing-scripts/intro-to-scripts/#execution-order-of-scripts.

37. POSTMAN, INC. *Store and reuse values using variables* [online]. 2024. [cit. 2024-04-24]. Dostupné z: https://learning.postman.com/docs/sending-requests/variables/variables/.

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `doc` ..... dokumentace systému
    - `sinis` ..... zdrojový kódy systému
    - `test` ..... zdrojový kódy testů
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF