Mendelova univerzita v Brně
Provozně ekonomická fakulta

Informační systém pro nájemní bytové domy

Bakalářská práce

Vedoucí práce:
Ing. Oldřich Faldík

Tomáš Jurníček

Brno 2016

Chtěl bych poděkovat především svému vedoucímu bakalářské práce Ing. Oldřichu Faldíkovi za jeho vedení, odborné rady a také za trpělivost a čas, který mi věnoval v průběhu celého řešení. Dále chci poděkovat Jiřímu Strouhalovi za zprostředkování tématu, jeho nápady a ochotu při testování výsledného informačního systému.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Informační systém pro nájemní bytové domy** vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 17. května 2016 .....

Abstract

Jurníček, T. Information System for rental apartment buildings. Bachelor thesis. Brno: Mendel University in Brno, 2016.

The aim of this thesis is to create a registration system for the administration of an apartment building. Its functionality is based on the specific needs of a particular owner and is formed as a web application. The first phase of the solution involved collecting of all functional and non-functional system requirements, followed by an analysis of the current situation and some of the suitable existing solutions. The next phase consisted of designing particularly the structure of the information system, the graphical interface and the database. The resulting solution was implemented within the Nette framework in combination with selected tools and the MySQL database, tested and then deployed to the production environment.

Keywords

Information system, web application, rental house, tenement house administration, Nette framework, Bootstrap, MySQL, Javascript.

Abstrakt

Jurníček, T. Informační systém pro nájemní bytové domy. Bakalářská práce. Brno: Mendelova univerzita v Brně, 2016.

Cílem této bakalářské práce je vytvořit evidenční informační systém pro správu nájemního domu. Jeho funkcionalita vychází ze specifických potřeb konkrétního majitele a je vytvořen ve formě webové aplikace. V první fázi řešení byly shromážděny veškeré funkční a nefunkční požadavky na systém, poté došlo k analýze současného stavu a výsledného zhodnocení z hlediska existujících řešení. Následovala fáze návrhu především struktury informačního systému, grafického rozhraní a databáze. Výsledné řešení bylo implementováno ve frameworku Nette v kombinací s vybranými nástroji a databází MySQL, otestováno a následně nasazeno na produkční prostředí.

Klíčová slova

Informační systém, webová aplikace, nájemní dům, správa nájemního domu, framework Nette, Bootstrap, MySQL, Javascript.

Obsah

1 Úvod a cíl

1.1 Úvod

Moderní technologie a jejich rozvoj neustále zvyšují vliv v drtivé většině lidských činností. Dnešní informační doba nám umožňuje efektivně zpracovávat data a využít je v příslušné reprezentaci takovým způsobem, který může sloužit pro její další analýzy či zpracování. Fyzicky posbíraná data mají tedy velký nedostatek v tom, že není možné je tak snadno interpretovat v tolika různých podobách, kromě toho jsou časově omezená a jejich uspořádání je značně limitováno.

Ať už to jsou velké podniky či menší výdělečné subjekty, v každém konkrétním případě existují informace, které je třeba evidovat a stejně tak i operace, které se dají zefektivnit za pomoci výpočetní techniky. S klidným svědomím mezi tyto subjekty můžeme zařadit také majitele nájemních domů. Jejich role je často velmi různorodá. Spravují dům, jednají s nájemníky, zařizují veškeré opravy či údržby bytů či společných prostor, řeší výpočty spotřebovaných energií, eviduje vybavení bytů, platby, uzavírají nájemní smlouvy a spousty dalších činností, které zaručují bezproblémový chod nájemního domu.

Z těchto činností vzniká majiteli nemálo povinností, které vyžadují pozornost a velmi často i spolupráci nájemníka. Řešením je vytvoření vhodného informačního systému, které podle vymezených kritérií umožní evidenci veškerých součástí nájemního domu a odlehčí od administrativních úkonů.

1.2 Cíl

Cílem práce je analyzovat požadavky a procesy, které vyžaduje konkrétní majitel nájemního domu. Na základě shromážděných požadavků poté navrhnout a nasadit moderní informační systém, postavený na nejnovějších webových technologiích. V prvních fázích řešení je nutné prozkoumat současný software, který v této oblasti trh nabízí. Na základě průzkumu se poté lze snáze zaměřit na silné stránky či nedostatky, jenž hotové varianty obsahují a dle dalších analýz přejít k návrhu a vývoji vlastního řešení, které bude zcela vyhovující svou funkčností a rozsahem.

2 Nájemní bydlení

Informace uvedené v této kapitole zohledňují možné změny právní úpravy podle nového občanského zákoníku (zákon 89/2012 Sb. platný od 1. ledna 2014.)

2.1 Nájem nemovitosti

Obecně je úprava nájmu definována jako úplatný právní vztah, sjednaný na vymezenou dobu (určitou či neurčitou). Představuje úplatu, která je odměnou za dočasné užívání individuálně určené věci nebo její části. Platnost dohody mezi stranou pronajímatele a nájemníkem je dána nájemní smlouvou. (Ronovská, 2012, s. 18-21)

2.2 Nájemní smlouva

Nájemní smlouva je založena na smluvní volnosti obou stran, což umožňuje dle zákona stanovit téměř jakoukoliv dohodu s ohledem na dobré mravy a postavení osob. Povinnost pronajímatele i nájemníka by měly být vyvážené. Nájemní smlouvy se vyhotovují ve dvou stejnopisech. Každá smluvní strana obdrží jeden výtisk a stvrdí účinnost smlouvy svým podpisem. Smlouva musí obsahovat popis předmětu (byt či dům), výši nájemného a dobu platnosti. (Ronovská, 2012, s. 22-26)

2.3 Nájemné

Od roku 2006 nabyla platnosti právní úprava (zákon č. 107/2006 Sb.), která řeší nájemné. Nahradila tím zákon č. 40/1964 Sb., který do té doby dovoloval Ministerstvu financí regulovat výši nájemného, po tomto datu je již výše nájmu otázkou dohody mezi oběma smluvními stranami. Tato novela zákoníku přinesla také úpravy práv a povinností účastníků nájemního vztahu. (Finance, 2016)

2.4 Práva a povinnosti pronajímatele bytu

Pronajímatel je především:

- povinen odevzdat byt ve stavu způsobilém – funkční vybavení, vymalováno (alespoň základní nátěr), provedené drobné opravy
- povinen zajistit nájemníkovi plný a nerušený výkon práva užívat byt
- povinen odstranit závady v řádném užívání bytu
- oprávněn požadovat úplatu za užívání bytu
- oprávněn požadovat složení kauce za užívání bytu - zákonem je vymezena maximálně částka trojnásobku měsíčního nájemného
- oprávněn vstoupit do bytu v zákonem daných případech (např. kvůli opravám)

- oprávněn ukončit nájem výpovědí či dohodou

(Ronovská, 2012, s. 27-28)

2.5 Práva a povinnosti nájemníka bytu

Nájemník je především:

- oprávněn užívat byt řádně
- povinen platit nájemné a jiné platby spojené s užíváním bytu řádně a včas
- povinen oznámit pronajímateli změny v počtu osob v bytě
- povinen v případě dlouhodobé nepřítomnosti stanovit kontaktní osobu
- povinen upozornit pronajímatele na potřeby oprav
- povinen umožnit vstup do bytu
- povinen umožnit pronajímateli instalaci a údržbu zařízení bytu

(Ronovská, 2012, s. 28-30)

3 Současný stav

Nájemní dům, o kterém se zde pojednává, je střední velikosti a obsahuje 8 samostatných bytových jednotek a další nebytové prostory. Nachází se na adrese *Partyzánská 19* ve městě Prostějov (Olomoucký kraj). V jednom z bytů žije i samotný pronajímatel, což umožňuje snazší komunikaci s nájemníky.

Pronajímatel v současnosti eviduje složky v písemné podobě s daty o každém bytě. Veškeré změny týkající se nájemníků a provozu nájemního domu je nutné archivovat, s tímto úkolem se pojí značná míra administrativní zátěže. Ve velkém množství dokumentů je obtížné se orientovat a dohledávat starší data. Jediné dokumenty vedené v elektronické podobě jsou nájemní smlouvy, které si pronajímatel sám dle potřeb aktualizuje. Další kategorií jsou účetní výkazy, které jsou předávány externí účetní, která na jejich základě vytváří daňová přiznání.

Majitel nájemního domu každý rok musí vytvářet tzv. „vyúčtování vodného a stočného“, což je výpočet spotřeby vody, kterou konkrétní nájemník za sledované období spotřeboval. Jelikož chodí vyúčtování globálně pro celý nájemní dům, je třeba sledovat stavy měřidel, které jsou zabudovány pro každý byt zvlášť. Tyto stavy je třeba zaznamenávat i u plynu a elektřiny a společně sepsané hodnoty poté každý měsíc odevzdávat pronajímateli do evidence. Komunikace mezi pronajímatelem a nájemníky je nedostačující, nájemníci si často zapomínají evidovat data ze stavů měřidel.

Historická údaje o nájemnících, bytech, vybaveních a dalších položkách jsou velmi těžko dohledatelné a představují jednu z velkých slabin současného fungování celého nájemního domu. Majitel nájemního domu má navíc velmi časté tzv. „mimořádné výdaje“ (opravy, stavební práce, údržby zařízení), které nelze zpětně dohledat.

3.1 Funkční požadavky na systém pro správu nájemního domu

Z analýzy současného stavu a zejména z potřeb majitele nájemního domu vyplynuly tyto funkční požadavky:

- evidence základních součástí nájemního domu a možnost jednotlivé údaje editovat, přidávat či mazat
- propojení privátní administrace s veřejně dostupnou webovou prezentací, kde bude možné přidávat veřejná oznámení
- možnost přihlášení nájemníků do systému s omezeným přístupem
- jednoduchá práce s informačním systémem a účelné uspořádaní jednotlivých modulů, které umožní intuitivní práci
- vytváření finanční bilance - porovnání příjmových a výdajových plateb, grafické znázornění

- požadavek webové aplikace
- nízké pořizovací náklady

3.2 Nefunkční požadavky na systém pro správu nájemního domu

Dle uvážení byly poté přidány tyto nefunkční požadavky:

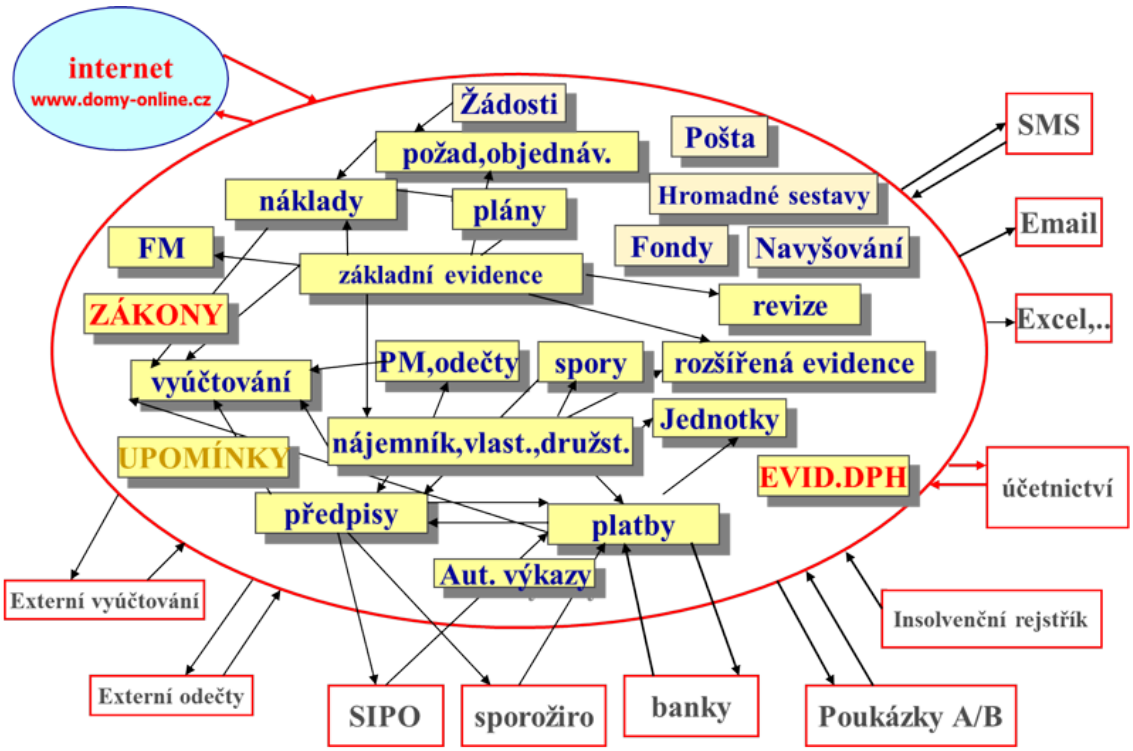
- možnost připojení se k systému odkudkoliv s pomocí jakéhokoliv zařízení prostřednictvím Internetu – responzivní přístup
- autentizace a autorizace uživatele přihlašujícího se do systému
- přiměřená úroveň zabezpečení a ochrany dat
- rozšiřitelnost a modifikovatelnost pro případné budoucí změny
- spolehlivost a rychlost systému – důraz na správnou funkci a minimální odezvu při práci v systému

3.3 Existující systémy podporující správu nájemních domů

Dnešní doba nabízí softwarová řešení pro různé oblasti lidské činnosti, stejně tak nalezneme několik variant určených pro potřeby majitelů nájemních domů. Výběr reprezentantů současného sortimentu byl proveden zejména na základě odlišných přístupů k velikostem a rozšiřitelnosti samotného systému. Jelikož se liší také stáří produktů, které jednotlivé společnosti nabízí, je možné sledovat měnící se požadavky uživatelů v průběhu několika let. Ve většině variant vybraných programů se jedná o jakýsi balíček služeb, který umožní uživateli spravovat základní i rozšířené údaje o jednotlivých bytech, nájemnících, platbách a podobně. Některé funkce mohou být velmi užitečné a přínosné, jiné zase překážející.

3.3.1 WinDomy

Jedná se o software od společnosti *O.K. – Soft* a nabízí systém pro správu nájemních domů s různým typem prostor – nájemní, vlastnické, garáže, družstevní byty a další. Je jedním z nejrozšířenějších řešení v této oblasti a v současnosti jej využívá více než 450 firem, které s jeho pomocí spravují více než 350 000 prostor po celé republice. Je dodáván primárně jako desktopová aplikace, ovšem nabízí přístup i přes webové rozhraní *http://www.domy–online.cz*. Mezi přednosti tohoto softwaru patří především to, že pomocí modulové skladby si uživatel koupí jen to, co potřebuje. Samotná cena softwaru se variabilně mění podle počtu spravovaných prostor, počtu modulů a instalací programu. Nejlevnější varianta stojí 3977 Kč bez DPH. (Oksoft, 2016)



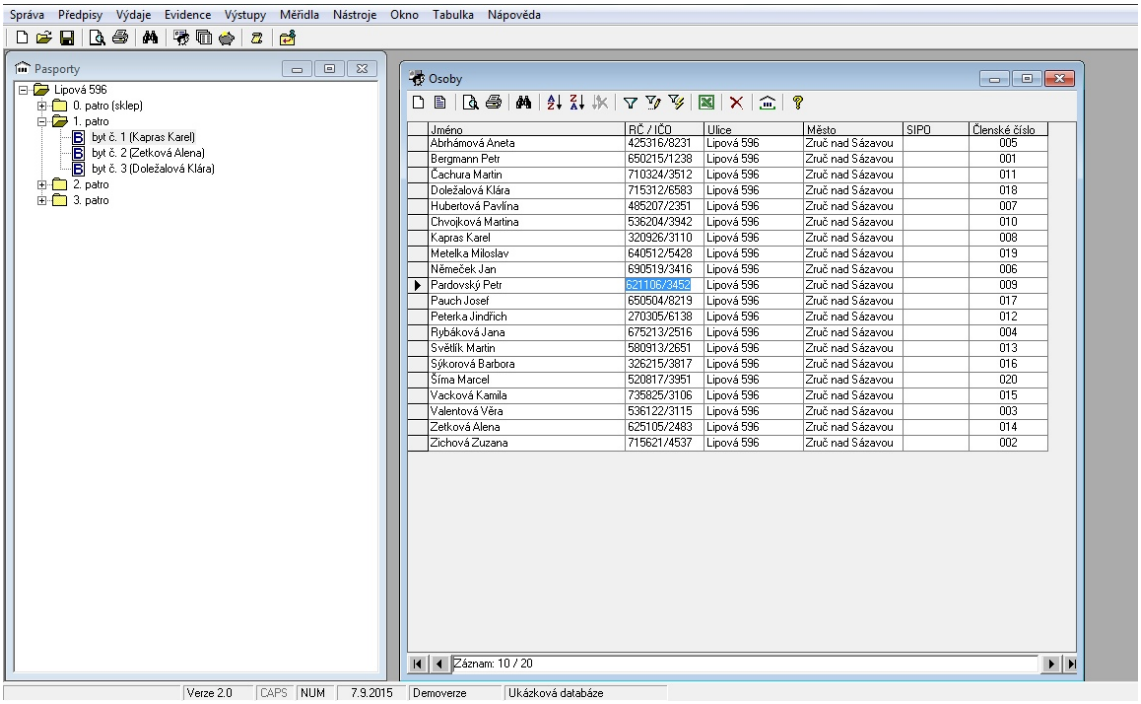
Obrázek 1: Schéma aplikace WinDomy (Oksoft, 2016)

Zhodnocení s ohledem na požadavky:

Aplikace jako celek umožňuje mnoho nastavení. Na první pohled působí dost chaoticky. Prostředí je graficky velmi neutrální a ničím výrazně nezaujme. Tato varianta poskytuje bohaté možností v oblasti evidence jednotlivých údajů, také je možné využít ji prostřednictvím webu. Ostatní funkční požadavky zde nejsou splněny.

3.3.2 SBN – správa bytů a nemovitostí

Software od společnosti *VYDAS – software s.r.o.*, který je určen pro subjekty spravující nájemní byty a nemovitosti. Jeho výhodou je zejména to, že je přizpůsobený pro účetní programy (*POHODA*). Program je tentokrát dodáván pouze v desktopové verzi a je na výběr z několika placených licencí. U varianty *LITE* je možno pracovat pouze s jednou databází na jednom počítači, případně produkt *STANDARD*, který nabízí větší možnosti. Nejlevnější varianta do 25 jednotek (bytový či nebytový prostor) vyjde 2 541 Kč s DPH. Kromě toho lze dokupovat různé služby a rozšíření. (Vydas, 2016)



Obrázek 2: Ukázka pracovního prostředí z SBN

Zhodnocení s ohledem na požadavky:

Daný software je orientován spíše k účetnictví, což nespadá do požadavků na nájemní systém. Uživatelské prostředí je poněkud strohé a samotné volby jsou sice uspořádané v horním menu, kde by mělo být vše po ruce, i přesto se s programem příliš intuitivně pracovat nedá. Ačkoliv obsahuje dostatek funkcí, není brán ohled na jistý uživatelský komfort při práci s více okny. Tato varianta zcela odporuje primárním funkčním požadavkům.

3.3.3 Bytová agenda

Dalším softwarem pro majitele bytových domů, bytová družstva či realitní firmy je *Bytová agenda*, který v současnosti využívá přes 100 subjektů. Skládá se z několika celků. Prvním z nich je *nájemné* sledující platby, vyúčtování a fakturace. *Pasporty* obsahují především údaje týkající se samotných bytů a jejich nájemníků/vlastníků. „Vyúčtování služeb“ eviduje náklady, umožňuje tvořit rozúčtování. Systém umožňuje více možností přístupu – desktopový i webový. (Sysag.cz, 2016)

Zhodnocení s ohledem na požadavky:

Bytová agenda je poměrně zajímavá varianta, která nabízí uživateli značné možnosti, ale většině funkčních požadavků není vyhověno. Funkce jsou poměrně přehledně uspořádané. Prostředí však neodpovídá aktuálním uživatelským požadavkům.

3.3.4 Building manager

Program je dílem společnosti *IC software* a slouží pro potřeby související se správou nemovitosti, včetně ekonomických a účetních agend. Opět se skládá z několika modulů, které dohromady tvoří bohatý přehled o součástech bytového domu (nájemníci, platby, hospodaření, byty). Zajímavým prvkem je fotodokumentace technického stavu nemovitosti. Disponuje velkým prostorem pro účetní operace a vyúčtování. Produkt lze zakoupit ve verzi *Lite* za 28 000 Kč bez DPH. Tato varianta obsahuje téměř všechny moduly, ale je omezena pouze pro jednoho uživatele. Ve víceuživatelských verzích pak postupně cena prudce narůstá. Co se týče typu programu, jedná se o čistě desktopovou verzi. (Icsoftware, 2016)

Zhodnocení s ohledem na požadavky:

Building manager je poměrně dražší produkt, který je pouze v desktopové verzi. Kladené požadavky na systém jsou z velké části nesplněny. Program sice nabízí několik zajímavých funkcí rozšiřujících použitelnost programu, ovšem pro konkrétní požadavky zadavatele nejsou relevantní.

3.4 Závěrečné zhodnocení současných softwarových řešení

I když existují zajímavá funkční řešení, jejich služby jsou ve většině případů omezené a těžko rozšiřitelné pro konkrétní požadavky zadavatele. Zároveň je také důležité si uvědomit, že větší softwarové společnosti si za dané řešení účtují poměrně vysokou částku, což pro každého nemusí být zcela optimální, zvlášť pokud se jedná jako v našem případě o uživatele, který vlastní několik bytů či samotnou nemovitost.

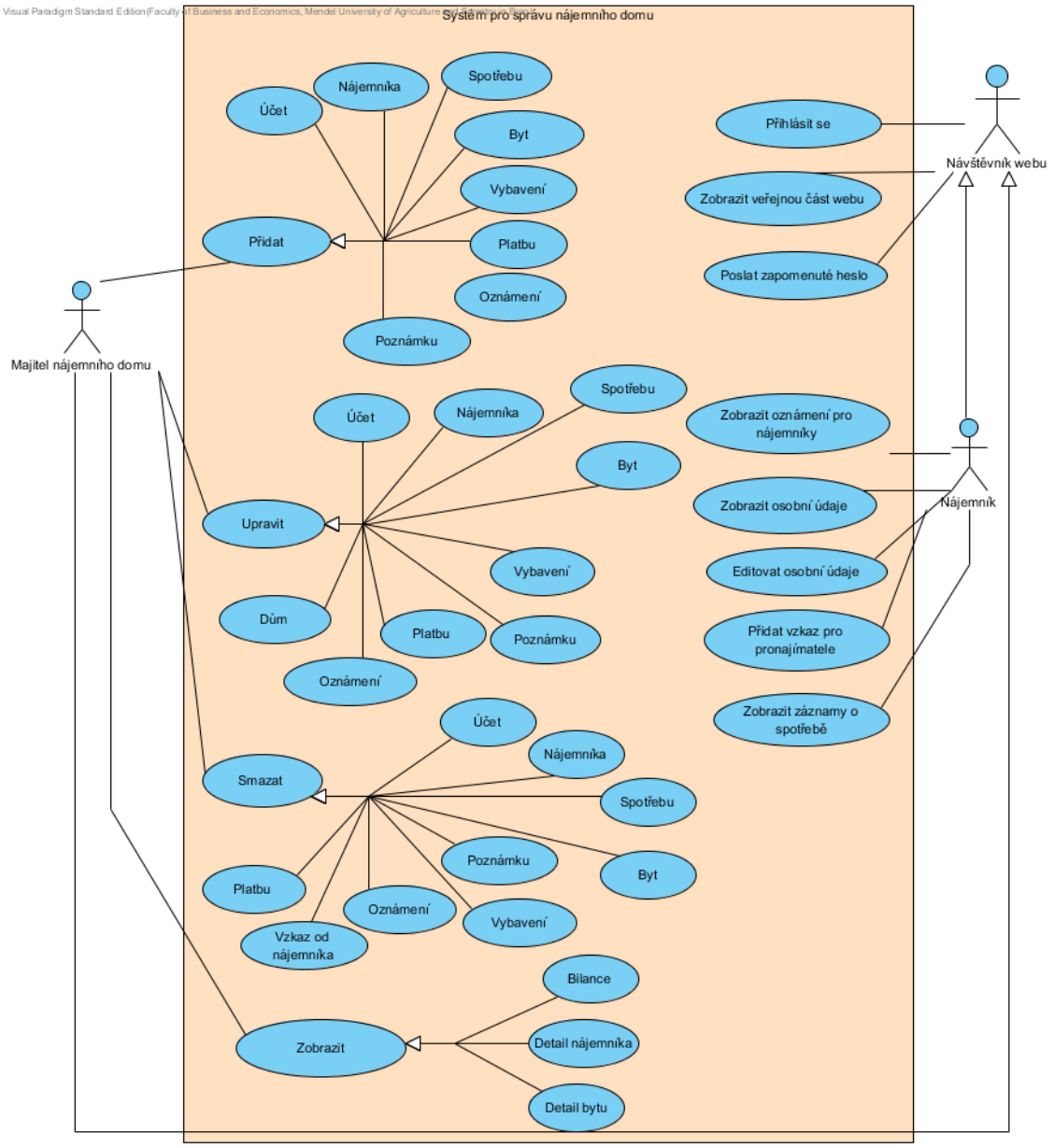
Dalším velkým problémem současných systémů tkví především v neaktuálnosti. Mnoho z nich totiž bylo vytvořeno před několika lety a za tu dobu nebylo příliš měněno. S tímto také souvisí fakt, že uživatelská prostředí jsou často velmi strohá, bez nápadu a při práci nenabízí mnoho uživatelského komfortu. Z toho důvodu pro potřeby majitele nájemního domu malého rozsahu nevyhovuje žádné z vybraných řešení. Dále zde žádná z variant nenabízí komunikaci mezi majitelem nájemního domu a nájemníky, jedná se sice o specifický požadavek, přesto je výhodný a originální, což dodá programu jistou přidanou hodnotu.

4 Návrh vlastního řešení

Z rešerše analyzující současný stav softwaru spravujícího nájemní domy vyplynulo, že v současnosti neexistuje žádné dostatečně vhodné hotové řešení, které by obsáhlo specifické požadavky ze strany zadavatele. Východiskem je navrhnout takových informační systém, který bude šitý na míru a vyhovovat ve všech ohledech.

4.1 Diagram případu užití

Diagram případu užití neboli *Use-case diagram* je z hlediska návrhu informačního systému prvním krokem, který definuje aktory a samotné akce (USE-CASE), které mohou v systému dělat. Celý diagram je možné vidět dále na Obr. 3.



Obrázek 3: Diagram případu užití

V diagramu jsou vytvořeni 3 aktoři. První z nich je *Návštěvník webu*, který představuje jakéhokoliv uživatele, který vstoupí na veřejně přístupné stránky, odkud se bude moci dostat na přihlašovací stránku do samotného jádra systému. Z tohoto aktora vychází dědičnost na další dva potomky, z nichž jeden je *Nájemník* a druhý *Majitel nájemního domu*. Pro každého z nich jsou poté definovány akce, které mohou provádět. V případě *Nájemníka* je tento výčet nesrovnatelně chudší, než v případě *Majitele nájemního domu*.

4.2 Návrh databázového schématu

Dalším krokem je tvorba návrhu databáze. Tato fáze probíhala v několika iteracích, kdy se k několika základním prvkům přidávaly některé další. Byl vytvořen logický model, jehož základními stavebními kameny jsou *entity*, *atributy* a *vazby*. Diagram znázorňující výsledek návrhu je z důvodu přehlednosti rozdělen do dvou samostatných obrázků, které jsou uvedeny níže na Obr. 4 a Obr. 5.

4.2.1 Popis jednotlivých entit

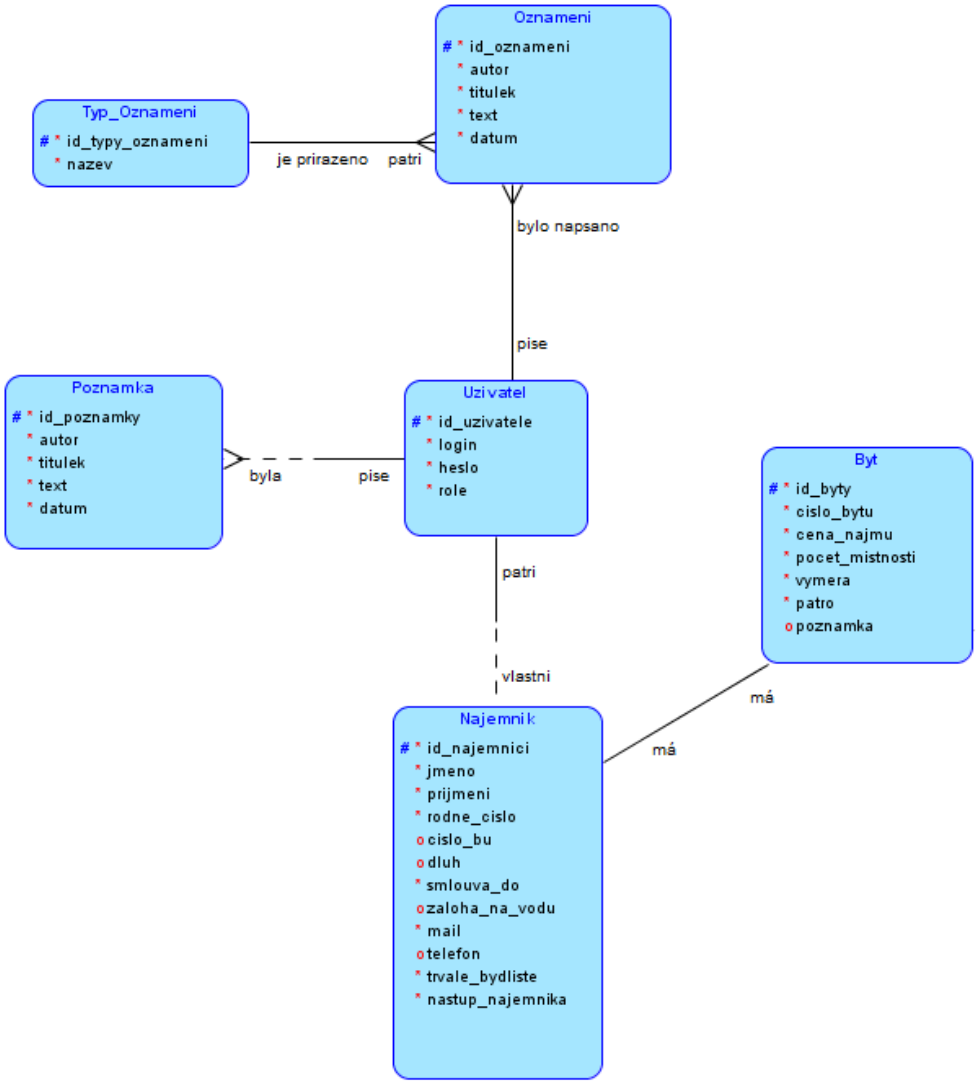
- **Byt**
  Entita *Byt* tvoří srdce celého návrhu a vytváří většinu klíčových vazeb. Právě zde se nachází základní informace o jednotlivých bytech. Velmi důležitý údaj představuje `cislo_bytu`, které by mohlo navodit dojem, že je vytvářeno genericky, ale není tomu tak. Majitel domu má byty označené čísly dle svého vlastního uvážení a toto číslo se poté uvádí v nájemních smlouvách a dalších dokladech. V této entitě nalezneme také další technické parametry jako `pocet_mistnosti`, `vymera`, `patro`. Je zde uvedena položka `cena_najmu`, která stanovuje výší nájemného pro nájemníky v zadaném bytě.

- **Nájemník**
  Tato entita obsahuje nejvíce položek. Kromě základních údajů o nájemnících jako `jmeno`, `prijmeni`, `rodne_cislo`, `cislo_bu`, `telefon`, `mail`, `trvale_bydliste`, najdeme i další údaje týkající se časovému vymezení. Nejdůležitější z nich je údaj `smlouva_do`, který představuje datum, ke kterému dochází ke konci platnosti nájemní smlouvy. Dále je uchována informace o tom, kdy se nájemník nastěhoval v položce `nastup_najemnika`.

- **Uživatel**
  Tato neméně důležitá entita je navržena zejména z důvodu rozlišení různých typů uživatelů v systému. Najdeme zde údaje, které slouží k přihlášení `login` a `heslo`. Pro rozdělení pravomocí v systému poslouží údaj `role`, na jejímž základě je možné od sebe oddělit uživatelské skupiny. První z nich, s maximální úrovní oprávnění bude sloužit pro samotného majitele nájemního domu. Druhá skupina bude mít značně omezené možnosti a poslouží pro nájemníky.

- **Oznámení**
  Oznameni slouží pro vytváření sdělení, které bude rozlišeno dle svého primárního klíče. Obsahuje v sobě informace jako `titulek`, `text`, `autor` a `datum`, kdy bylo oznámení vytvořeno.

- **Typ oznámení**
  Je jednoduchá entita, která slouží pouze k vytváření více typů k entitě *Oznameni*. Její primární klíč `id_typy_oznameni` pak představuje číselnou hodnotu, ke které je přiřazena položka `nazev`.
- **Poznámka**
  Je entitou velmi podobnou *Oznameni*, přesto poslouží k jinému účelu. Obsahuje také atributy `titulek`, `text`, `datum` a `autor`.



Obrázek 4: Schéma databáze první část

- **Dům**
  Entita *Dum* je postavena ve stromě evidovaných údajů na nejvyšší úroveň. Ukládá v sobě informace, které se týkají celého konkrétního nájemního domu. Především zde patří popisné údaje jako `mesto`, `ulice`, `cislo_popisne`, `psc`. Dále jsou uvedeny stavy energií v položkách `vodomer` a `elektromer` a také položku `datum_revize`, což je informace o tom, kdy byla naposledy vykonána revize domu, která probíhá v pravidelném časovém intervalu.

- **Vybavení**
  Entita *Vybaveni* slouží k evidenci obsáhlé součásti celého systému, tedy vybavení, které se nachází v každém z bytů. O každém kusu se evidují obecné informace jako `nazev`, `cena_porizeni`, `datum_porizeni`. Dále je zde odkaz na byt, ke kterému patří a samozřejmé také na typ vybavení, které je definováno dále.

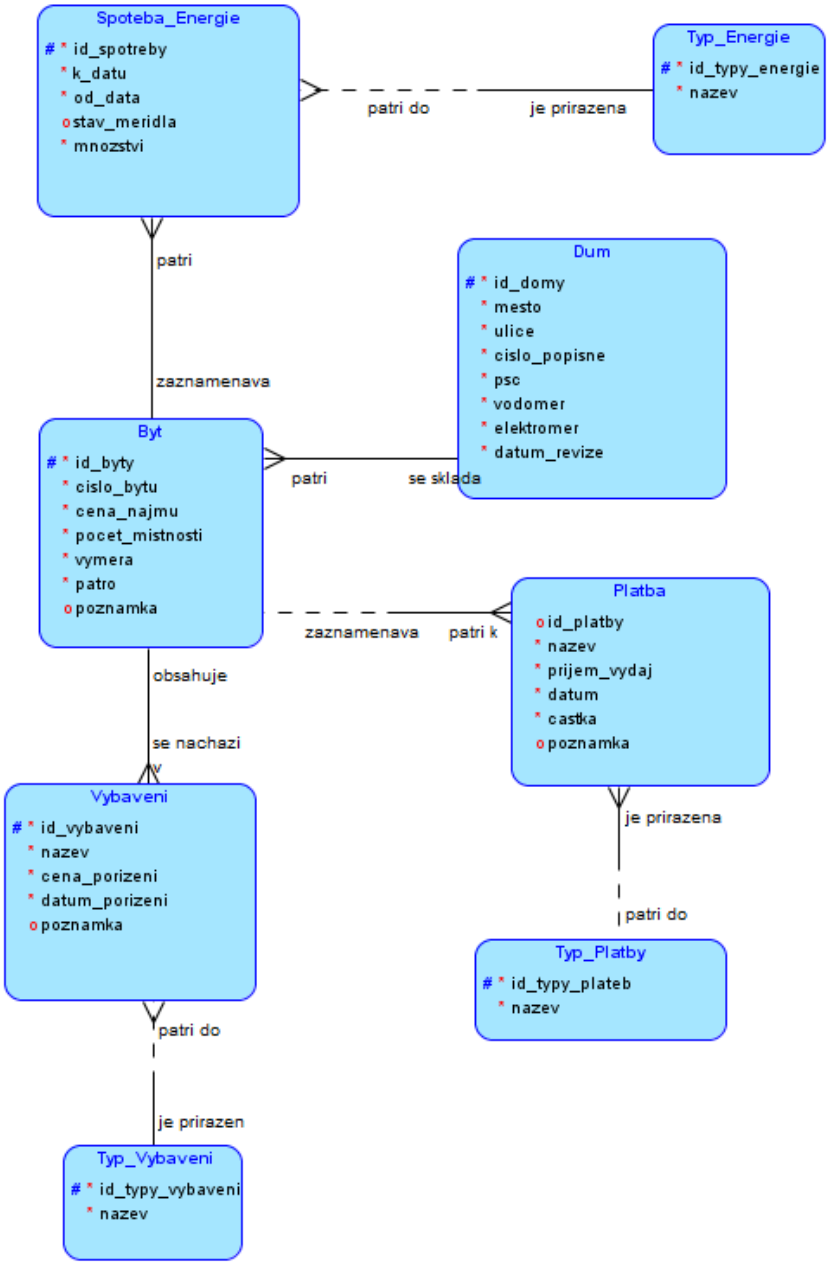
- **Typy vybavení**
  Zde najdeme jednoduchou strukturu, kde je definován primární klíč v kombinaci s položkou `nazev`.

- **Platba**
  Platby je další velmi obecnou entitou, která je v návrhu zejména pro sledování a uchovávání peněžních toků, které v nájemním domě probíhají. Opět jsou zde uvedeny základní položky pro rozlišení `nazev`, `datum`, `castka`. Důležitý údaj nalezneme v `prijem_vydaj`, který může nabývat pouze dvou hodnot. První z nich bude představovat platby, které mají výdajový charakter, tedy budou pro majitele nájemního domu položkami, které musí zaplatit. Naopak druhá zmíněná hodnota bude představovat platby příjmového charakteru, která indikuje profit.

- **Typ platby**
  Entita představuje kategorie plateb, které se mohou v nájemním domě vyskytovat. Mohou to být například platby za údržbu či opravy, účetní či právní úkony a mnoho dalších.

- **Spotřeba energie**
  Entita *Spotreba_energie* představuje obecnou strukturu, ve které budou uchovány veškeré záznamy týkající se spotřebovaných energií za jednotlivé byty. Základem je uchování dvou časových údajů, které vymezí období, v němž ke spotřebě došlo. K tomu slouží položky `od_data` a `k_datu`. Dále je nutné mít pro kontrolu uvedené množství spotřebované energie v `mnozstvi` a pro kontrolu také stav měřidla v `stav_meridla`, které poslouží pro kontrolní výpočty a vzájemně spolu pozitivně korelují.

- **Typ energie**
  Tato entita slouží k rozdělení položek ze *Spotreba_energie* a je v podstatě definována pouze pro tři typy energií, které domácnosti využívají – *voda*, *plyn*, *elektřina*.



Obrázek 5: Schéma databáze druhá část

4.3 Struktura informačního systému

Vzhledem k již definovaným údajům, která je nutné evidovat, je možné určit strukturu webu, která bude rozdělena do několika samostatných logických modulů.

Výčet modulů systému dle uživatelských rolí:

- Návštěvník webu
  - **Domovská stránka** – veřejná část webu se základními informacemi a odkazem k přihlašovací stránce
- Nájemník
  - **Účet** – přehledy, operace nad vybranými údaji
- Majitel nájemního domu
  - **Dům** - rozcestník, informace o domu
  - **Byty** - operace, přehledy nad údaji o bytech
  - **Nájemníci** - operace, přehledy nad údaji o nájemnících
  - **Vybavení** - operace, přehledy nad údaji o vybavení
  - **Spotřeba energií** - operace, přehledy nad údaji o spotřebě energií
  - **Platby** - operace, přehledy nad údaji o platbách
  - **Oznámení** - operace, přehledy nad údaji o oznámeních
  - **Uživatelské účty** - operace, přehledy nad uživatelskými účty nájemníků
  - **Vzkazy** - přehled vzkazů z uživatelských účtů nájemníků
  - **Bilance** - přehled finančního hospodaření domu

4.4 MVC architektura

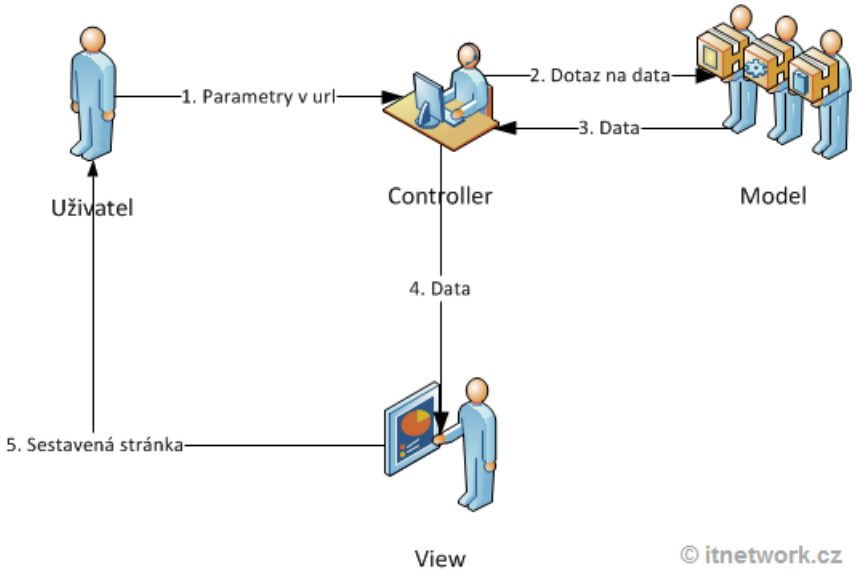
Je jednou z nových a rychle se rozvíjejících softwarových architektur, která se skládá ze tří základních komponent *Model*, *View*, *Controller*. Principem je oddělení logiky a samotného výstupu, což umožňuje jednodušší správu zejména rozsáhlejších aplikací.

Model se často označuje jako část architektury, která řeší veškeré logické operace. Patří mezi ně zejména manipulace s databází (dotazy), různé výpočty apod. Jeho úkolem není starat se o to, jaká data do něj přijdou a v jaké výsledné podobě budou interpretována.

View je další z trojice komponent, jejímž hlavním úkolem je interpretace dat uživateli na výstupu systému. Kromě toho však slouží také jako vstupní vrstva systému, jejímž prostřednictvím mohou uživatelé zadávat data, která poté pošlou

dále ke zpracování. Je postavena často na šablonovacím systému a využívá především značkovacích jazyků jako *HTML* a *CSS*.

Controller je propojovacím mostem mezi Modelem a View, který obsluhuje veškeré potřebné operace těchto dvou komponent. Prakticky to znamená, že zde dochází například k předání vstupních dat z formulářového prvku (*View*) a poté k jejich zpracování a odeslání k logickým operacím (*Model*). Princip celé architektury je zachycen na následujícím Obr. 6.(ITnetwork, 2016)



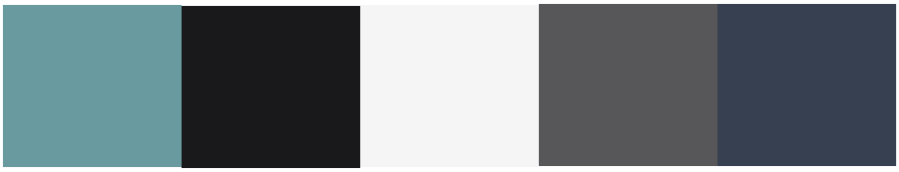
Obrázek 6: Princip MVC architektury (ITnetwork, 2016)

4.5 Návrh uživatelského rozhraní

Z požadavků na systém plyne několik důležitých doporučení na správné vytvoření uživatelského rozhraní. Je brán velký ohled na to, aby celá aplikace byla postavena logicky, přehledně a především intuitivně. To vše doplněné moderní a čisté grafické znázornění, která nebude obsahovat příliš rušivých elementů.

4.5.1 Barvy

Po zvážení několika variant bylo rozhodnuto o volbě barevného schématu, jehož základním konceptem je využití černé a bílé barvy, doplněné o modrou. Některé z využitých odstínů je možné vidět v následujícím Obr. 7.



Obrázek 7: Část použitých barevných odstínů

4.5.2 Logotyp

Pro daný informační systém je vždy vhodné zvolit ústřední prvek, který se bude vyskytovat na všech jeho podstránkách. Z toho důvodu bylo navrženo grafické logo představující dům skládající se z několika bloků, z nichž každý z nich znázorňuje byt.

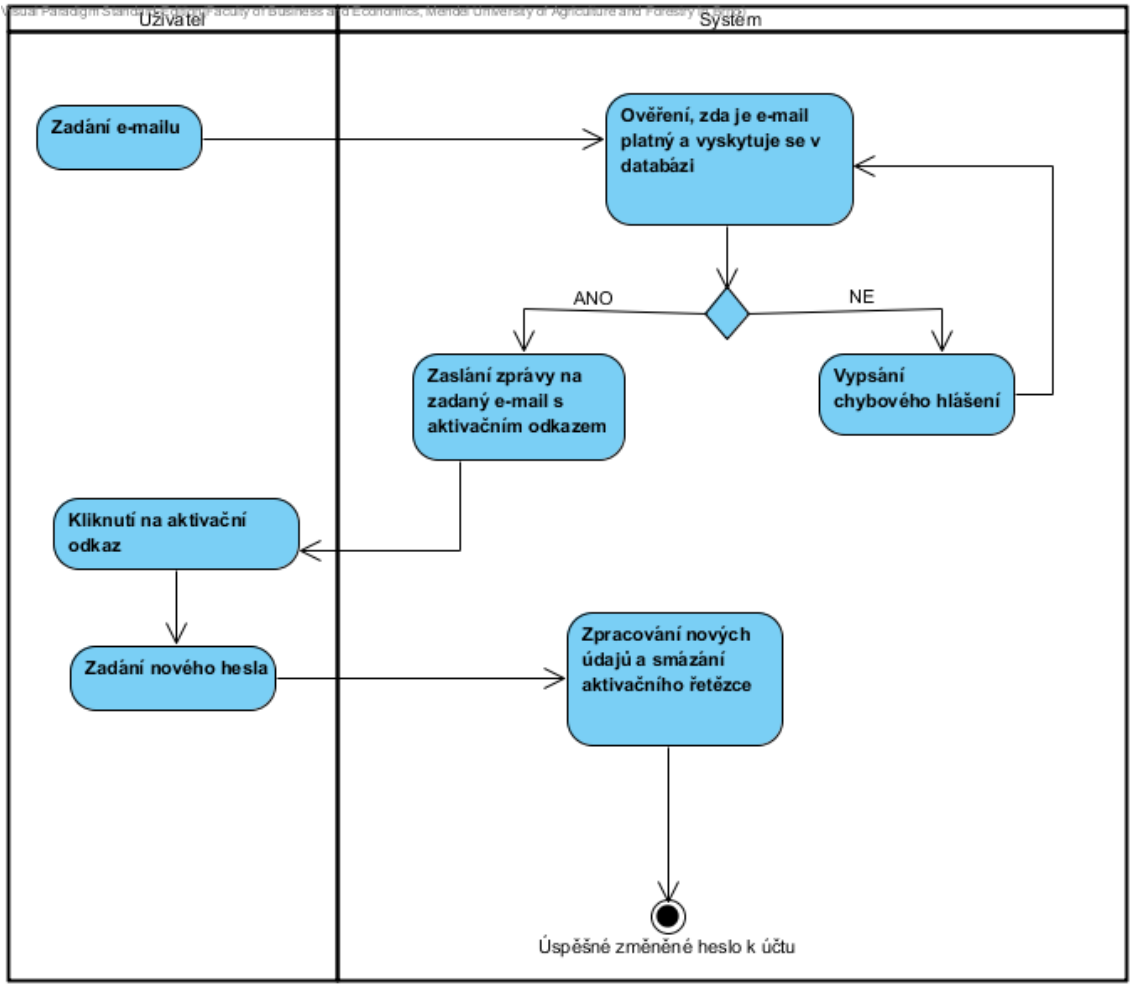
Obrázek 8: Výsledný logotyp znázorňující nájemní dům

4.5.3 Návrh přihlašovací stránky

Pro vstup do systému bude nutné projít stránkou s přihlašovacím formulářem. Podle uživatelského účtu, který má v sobě v databázi uvedenou roli, jsou uživatele rozděleni do příslušných skupin. Na základě toho také dojde po přihlášení nájemníka k přesměrování na jiný modul webu, než je tomu u majitele nájemního domu.

Posílání zapomenutého hesla

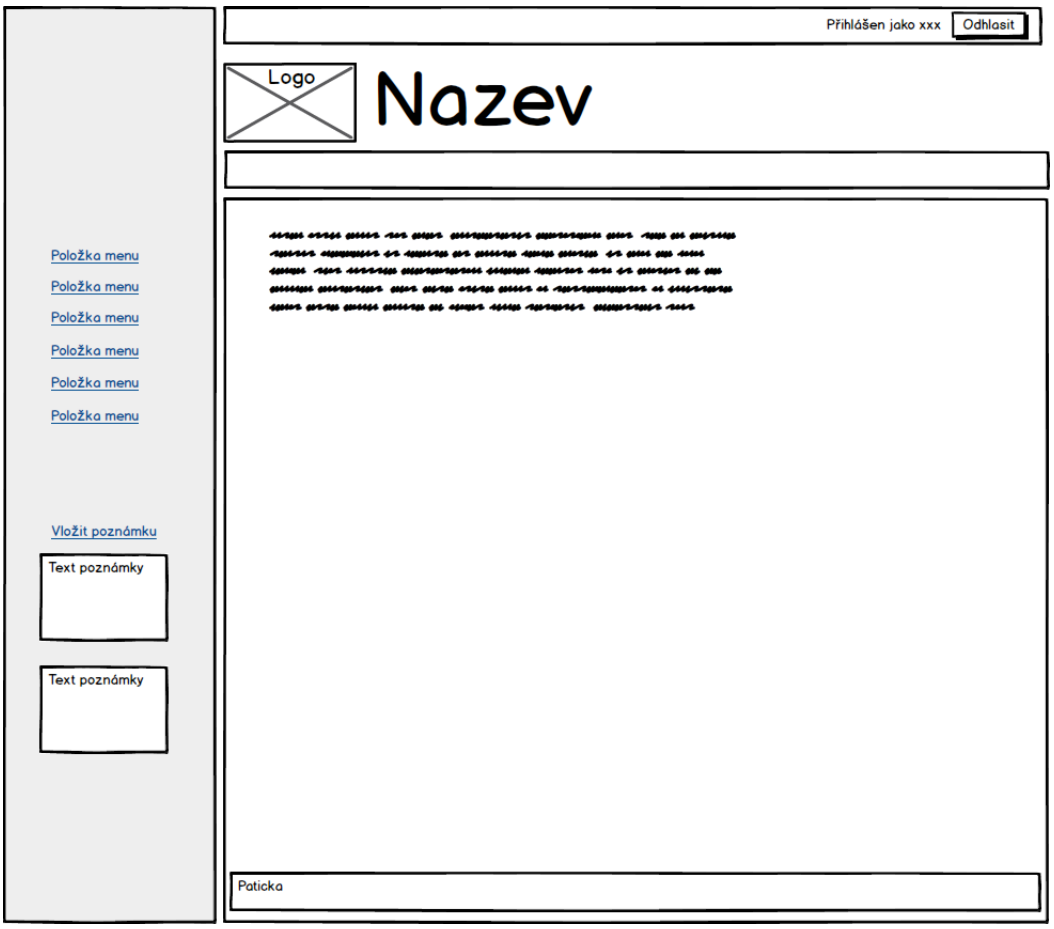
Jednou z důležitých funkcí, která se bude vyskytovat na přihlašovací stránce je právě možnost zaslat zapomenutého heslo. Je určena pro účty nájemníků, aby v případě, že zapomněli po nějaké době své heslo, mohli bez asistence majitele domu, vyřešit svá přihlášení. Princip této funkcionality je popsán v následujícím diagramu aktivit na Obr. 9



Obrázek 9: Diagram aktivit pro zasílání zapomenutého hesla

4.5.4 Návrh administrace pro přihlášené uživatele

Administrační rozhraní samotného webu musí být na první pohled přehledné a během několika málo sekund musí uživatel zřetelně rozpoznat, které možnosti v systému má. Hlavní navigace webu byla zvolena jako vertikální v levém bočním panelu. Samotná struktura stránky je poté standardně rozdělena v horní části obrazovky na hlavičku obsahující logo, název, rozcestník a tlačítko pro odhlášení ze systému. Dále následuje hlavní obsahová část, která zabere nejpodstatnější část celého okna. Ve spodní části se nachází jednoduchá patička s informacemi o webu a případnými odkazy. Výsledný návrh je možné vidět v Obr. 10.



Obrázek 10: Návrh administrace

5 Implementace

5.1 Vývojové prostředí

5.1.1 NetBeans

NetBeans je desktopové vývojové prostředí (IDE), která je především určené pro vývoj v jazyce *Java*, *C/C++*, mobilní a webové aplikace (*PHP*, *HTML*, *Javascript*, *CSS*). Umožňuje vytvářet rozsáhlejší projekty, které obsahují složitější adresářovou strukturu. Mezi jeho základní funkce patří zvýrazňování syntaxe, napovídání konstrukcí jazyka a automatická kontrola určitých syntaktických pravidel (např. chybějící středník či závorka). Toto vývojové prostředí má mnoho rozšíření a je možné si doinstalovat k softwaru také chybějící doplňky (pluginy). K vývoji byla použita verze 8.1. (NetBeans, 2016)

5.1.2 XAMPP

XAMPP (X – multiplatformní, A – Apache, M – MariaDB, P – PHP, P – Perl) je open-source software určený pro vývojáře, který slouží pro vytvoření lokální serverové simulace na vlastním počítači. Při vývoji bylo využíváno dvou z jeho služeb. První z nich je *Apache HTTP server*, který je potřebný pro webové projekty a zejména tedy pro jazyk *PHP*. (Apachefriends, 2016)

Dále bylo zapotřebí databázové služby *MySQL*, která v sobě obsahuje i nadstavbu v podobě *phpMyAdmin*. Ta je napsána v jazyce *PHP* a umožňuje základní operace nad daty, tabulkami, také přímé vkládání SQL dotazů. (PhpMyAdmin, 2016)

5.2 Použité technologie

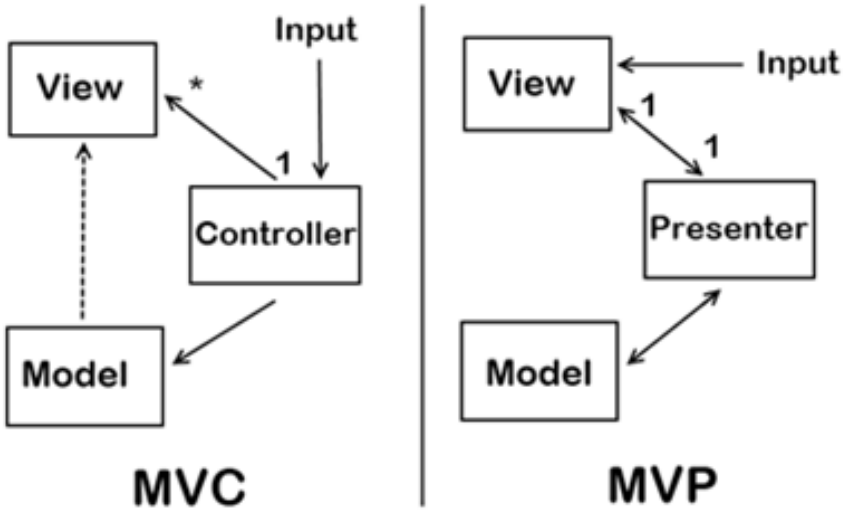
5.2.1 Framework Nette

Nette je PHP framework určený k tvorbě webových aplikací, jehož autorem je český vývojář *David Grundl*. Na jeho základech stojí v současné době několik velkých portálů jako *csfd.cz*, *uloz.to*, *slevomat.cz* a mnoho dalších. Byl také označen jako nejpopulárnější a nejvíce používaný framework v České republice. (Zdrojak, 2010) Obsahuje svůj vlastní šablonovací systém *Latte*, který nabízí mnoho příjemných pomůcek a nástrojů pro vývojáře. Součástí základních knihoven je také ladící nástroj *Tracy*, který umí na základě vzniklých chyb během vývoje vytvářet specifické hlášky, na jejíchž základě se snáze odlaďuje samotný kód.

Od verze *PHP 5* došlo k razantnímu zdokonalení výbavy objektově orientovaného programování, což je jeden z nejvýraznějších rysů této verze. Byly přidány dodatky jako explicitní konstruktory a destruktory, klonování objektů, abstraktní třídy, rozhraní, především však správa samotných objektů. Nette, stejně jako velká většina PHP frameworků staví právě na těchto objektových vlastnostech. (Gilmore, 2011, s. 29-30)

Porovnání MVP a MVC architektur

Vybraný Framework Nette je postaven na MVP architektuře, což je jakýsi derivát MVC, který upravuje některé principy komunikace mezi komponentami. Schématické rozdíly je možno vidět na Obr. 11.



Obrázek 11: Grafické znázornění vazeb mezi komponentami v MVP a MVC (Geekswithblogs, 2009)

MVP architektura je velmi podobná MVC a skládá se taktéž ze tří částí. Jsou jimi *Model*, *View* a *Presenter* (ten nahradil *Controller*). Na rozdíl od MVC, neexistuje vazba mezi *Model* a *View*, čímž je aplikace jednodušší a probíhá snadněji zejména její testování. Porovnání klíčových principů je popsáno v následující Tab. 1.

Tabulka 1: Srovnání základních rozdílů mezi MVC a MVP (Geekswithblogs, 2009)

MVC	**MVP**
Vstup probíhá přímo přes Controller.	Vstup je zadán pomocí View.
Vazba N – 1 mezi Controllerem a View.	Vazba 1 – 1 mapující View a příslušný Presenter.
View neví nic o Controlleru.	View udržuje reference na Presenter, ten reaguje na události z View. Tím pádem si je Presenter vědom View.
View má znalosti o Controlleru prostřednictvím Modelu.	View si není vědom Modelu.

5.2.2 MySQL

MySQL je open source relační databázový systém, který patří do kategorie DBMS (*Database Management System*). V současnosti je velmi rozšířeným a oblíbeným nástrojem, který pochází z dílny vývojářů Oracle. Stejně jako další alternativní databázové systémy (nejvíce podobnou alternativou je MariaDB), i MySQL využívá deklarativního programovacího jazyka SQL (*Structured Query Language*). Je možné jej využívat v kombinaci s technologiemi C/C++, Perl, PHP, Python, Ruby a mnoha dalšími. (MySQL, 2016)

Pro informační systém pro správu nájemního domu byl zvolen zejména z těchto důvodů:

1. Open source systém
2. Používaný již jako standard – spousty komerčních i nekomerčních projektů
3. Velká kompatibilita – nabízí ho drtivá většina webhostingových služeb
4. Jednoduchý – pro začátečníky je velice rychle využitelný
5. Je dostupný zdarma
6. Bezpečný – ochrana dat, podpora šifrovacích funkcí

(Novell, 2016)

Schéma databáze realizované v MySQL
Z logického modelu zmíněného v kapitole návrhu bylo možné zjistit stavbu entit, jejich atributy a vzájemné vazby. Na základě volby konkrétní technologie MySQL, bylo možné převést model do tohoto prostředí. Výsledkem je datový model, který řeší dále otázky konkrétně zvolených datových typů, jejich rozsah a další implementační požadavky. Schéma je možné vidět na Obr. 18.

5.2.3 CSS Framework Bootstrap

Bootstrap od společnosti *Twitter* je v současné době jednou z nejpoužívanějších knihoven pro definování vizuálního stylu prvků webové stránky. Je postavena na několika základních prvcích, které si poté může vývojář dále rozvíjet nebo je přímo používat. Velikou výhodou je responzivní chování a použitelnost ve všech prohlížečích bez nutnosti dodatečného ladění.

Mezi jedny z nejužitečnějších stylů, které Bootstrap nabízí, patří bezesporu *Grid systém*, který byl také použit pro realizaci popisovaného informačního systému. Tato komponenta je založena na jednoduchém principu, kdy definovaný blok je rozdělen do 12 sloupců, který je ve 4 různých rozlišeních (mobilní, tablet, desktopové a velké desktopové) možno přizpůsobit tak, aby zabraly určitý počet sloupců. Dále je v konkrétní implementaci využíváno formulářových prvků, tabulek, navigace, tlačítek, panelů a dalších typografických stylů. (Bootstrap, 2016)

Součástí Boostrap frameworku jsou také „*Glyphicons*“, což je knihovna 250 ikon, z nichž každá má svojí vlastní CSS třídu, tudíž jejich používání je velice jednoduché a intuitivní. Ikony lze navíc napasovat například i do tlačítek či nadpisu. Tato omezená sada je dílem českého grafika Jana Kovaříka a samostatně v mnohem bohatší podobě (v současnosti přes 800 ikon) je možné si ji zakoupit přímo na stránkách autora. (Kovařík, 2016)

Pro „rozhýbání“ některých prvků frameworku Boostrap existuje také Javascriptová část, z nichž nejvíce je v práci využíván tzv. „*Collapse*“, tedy funkce, která umí skrývat a znovu zobrazovat svůj vnitřní obsah.

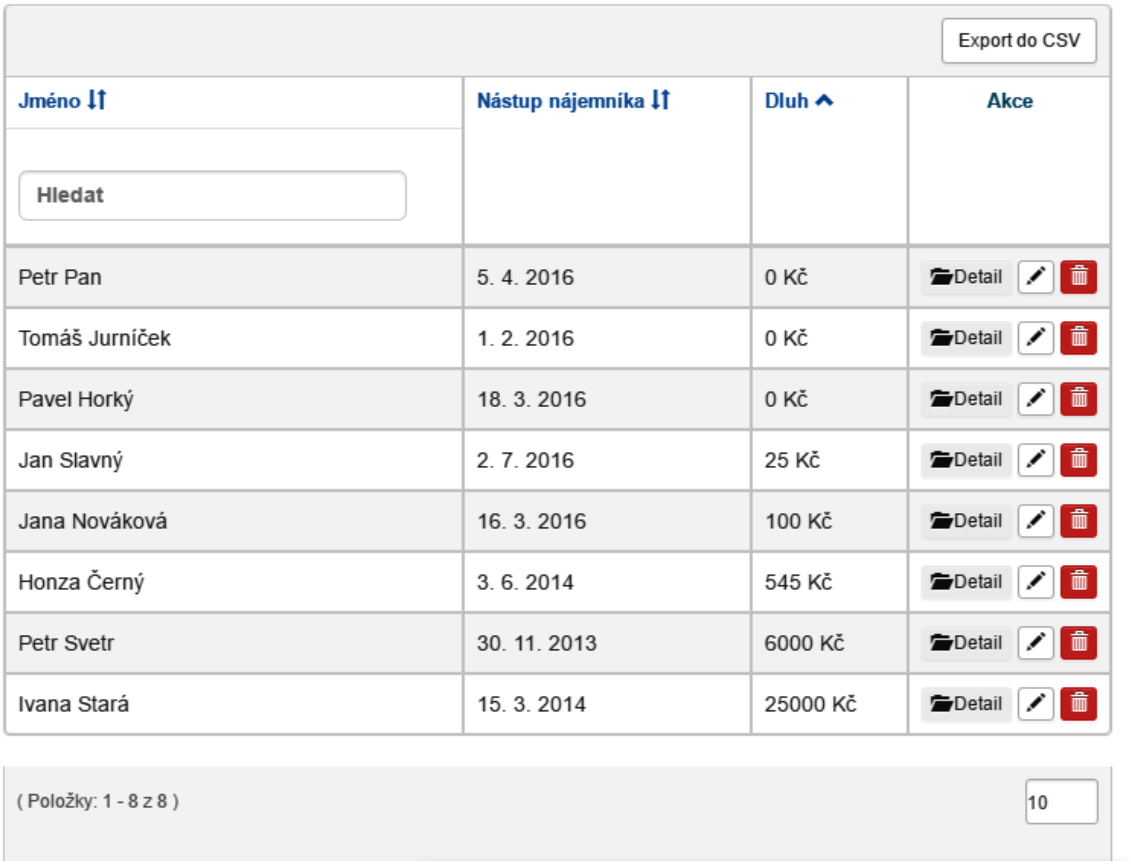
5.2.4 Google Charts

Jelikož v informačním systému v modulu *Bilance* je zapotřebí zobrazovat data v podobě grafů, bylo třeba zvolit technologii, která zajišťuje právě jejich vykreslování. Z několika možných variant bylo poté rozhodnuto, že bude využit *Google Charts*. Velkou výhodou je, že se jedná o skutečně velmi rychle použitelnou technologii, která obsahuje bohatou databázi různých typů grafů, které zcela vyčerpávajícím způsobem vyhovují kladeným požadavkům implementace.

Vytváření samotných grafů je poté záležitostí importu příslušných knihoven a dále javascriptového kódu, který je rozdělen do několika částí. První funkcí *google.load()* jsou definovány balíčky, které představují výčet využitých typů grafů (koláčový, sloupcový apod.). Následně ve funkci *google.setOnLoadCallback()* dojde k vytvoření zpětného volání, které naplní příslušná data. Poslední funkcí je poté *drawChart()*, která již definuje atributy grafů do příslušných proměnných a na konci je provolá ve funkci *google.visualization()*. (Google, 2016)

5.2.5 Ublaboo Datagrid

Navrhovaný informační systém se z velké části skládá z rozhraní pro práci s daty o jednotlivých evidovaných částech nájemního domu. Vzniká zde potřeba globálního přehledu nad samotnými položkami, jejich prohledávání, filtrování a řazení, stránkování. Právě z těchto důvodu byl využit tzv. „Datagrid“, který je samostatně vyvíjen k frameworku Nette jako doplněk. Konkrétně byl zvolen *Ublaboo Datagrid* od vývojáře Pavla Jandy. Jeho řešení vzniklo teprve začátkem roku 2016 a nabízí kromě potřebných funkcionalit i mnoho dalších, jenž umožní v budoucnu rozšiřovat i samotný informačního systému. (Janda, 2016)

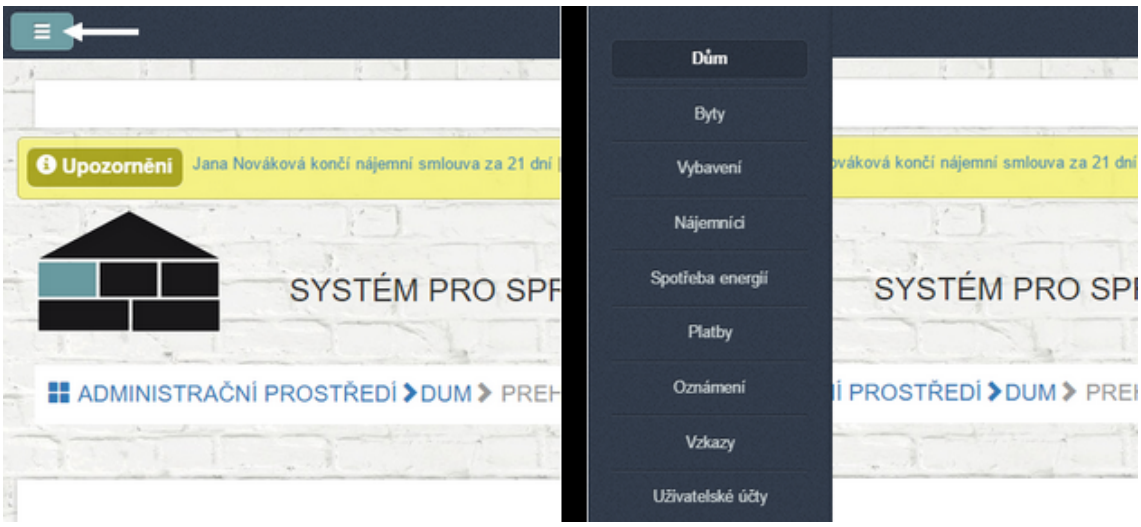


Obrázek 12: Ukázka použitého datagridu

5.2.6 Šablona Striped

Jako základ designu webové aplikace byla využita volně dostupná šablona (spadající pod licenci *The Creative Commons Attribution 3.0 License*), což umožňuje její volné používání pro komerční i osobní projekty. (HTML5 UP, 2016)

Tato šablona byla poté dále modifikována a rozšiřována dle potřeb. Její hlavní dominantou je vertikální menu v levém sloupci, které je zcela responzivní, což znamená, že při nižším rozlišení se celý sloupec defaultně skrývá a teprve pro najetí na ikonu tzv. „hamburger“ se menu opět objeví. Celá situace je znázorněna v následujícím Obr. 13.



Obrázek 13: Responzivní chování levného panelu s menu

5.3 Vybrané funkce systému

5.3.1 Autentizace uživatele

Ná základě existujících uživatelských účtů, může uživatel přistupovat do systému prostřednictvím přihlašovací stránky. Ve chvíli, kdy zadá svůj login a heslo, dojde k přesměrování do patřičného rozhraní na základě uživatelské role.

Uživatel má dále možnost zapamatování si přihlášení. Implementačně je toto řešeno ve funkci *formSucceeded()* v následujícím úryvku kódu.

```
if ($values->remember) {
$this->user->setExpiration('14 days', FALSE); //pamatovat prihlaseni
} else {
$this->user->setExpiration('60 minutes', TRUE); // nepamatovat prihlaseni
}
```

Obrázek 14: Přihlašovací rozhraní

5.3.2 Rozcestník pro práci s byty

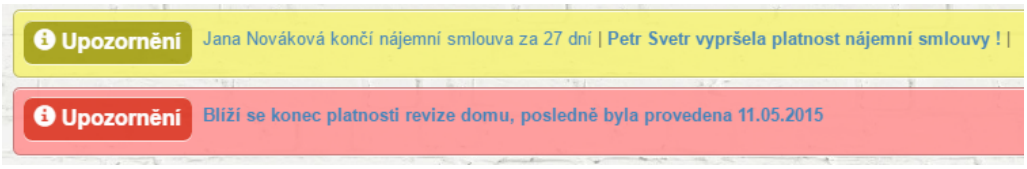
Po vzoru logotypu, který byl navržen jako dům skládající se z kostek, bylo vytvořen ústřední rozcestník, který slouží pro majitele nájemního domu ve chvíli, kdy potřebuje v rychlosti vybrat konkrétní byt a pracovat s údaji o něm.

Obrázek 15: Grafický rozcestník pro práci s byty

5.3.3 Upozornění

V administračním rozhraní majitele nájemního domu je vytvořena také funkcionalita upozornění, která varuje před končícími termíny vždy 30 dnů předem. V implementaci je to řešeno v SQL dotazu za pomocí podmínky `WHERE DATEDIFF(smlouva_do,CURRENT_DATE)<=30`. Celkově jsou navrženy dva typy upozornění:

1. Konec platnosti nájemní smlouvy
2. Konec platnosti revize domu



Obrázek 16: Upozornění umístěné v hlavičce webu

5.4 Bezpečnost

5.4.1 Oprávnění uživatelů

Bezpečnost aplikace z hlediska oprávnění přístupu uživatele na určitou stránku v systému, je řešeno za pomocí autorizátoru ACL (*Access Control List*), který framework Nette má již definovaný v třídě `Permission`. Princip spočívá v definování rolí, zdrojů a samotných oprávnění. Více je možné vidět v samotné ukázce z implementace.

```
$this->acl->addRole("admin"); //pridani role
$this->acl->addResource("byty"); //pridani zdroje
$this->acl->allow("admin", "byty", "vytvorit"); //opravneni
```

Ve chvíli, kdy správně nadefinujeme, kam který uživatel může vstoupit či nikoliv, můžeme se následně ptát skrze metodu *isAllowed()*, která v případě odpovídajícího nastavení ACL vrátí hodnotu *true* a umožní uživateli zobrazit obsah. Tento princip je využíván zejména v *BasePresenteru*, konkrétně v jeho konstruktoru, kde dochází k ověřování přístupových oprávnění.

5.4.2 Bezpečnostní rizika

- **SQL injection (vsouvání)**
  Jedná se o vsouvání neověřeného kódu do proměnné, která vstupuje do SQL dotazu. Ve chvíli, kdy nejsou data escapována, může docházet k provádění příkazů, které umožní útočníkovi získat přístup k citlivým datům. (Šilhavý, 2013, s. 70)

  Nette využívá při předávání parametrů do SQL dotazu konstrukce, která sama zajišťuje escapování proměnné, takže toto riziko eliminuje. Následující ukázka kódu objasní její využití. V prvním případě je proměnná vsouvána přímo do dotazu, což je velmi nebezpečně, druhá varianta využívá zástupného symbolu `?`, který je poté nahrazen odpovídajícím parametrem za zadaným příkazem.

  ```
  query("SELECT * FROM byty WHERE id_byty= $id_byty); // Nebezpecne
  query("SELECT * FROM byty WHERE id_byty=?", $id_byty); //Bezpecne
  ```

- **Cross-Site Scripting (Skriptování napříč stránkami)**
  Chyba využívající neošetřených výstupů, kdy lze podstrčit vlastní kód, čímž lze stránku modifikovat. V Nette frameworku je toto vyřešeno za pomocí technologie *Context-Aware Escaping*, která dané výstupy ošetřuje automaticky za pomocí escapování vypisovaných řetězců.(Nette Foundation, 2016)

- **Cross-Site Request Forgery (CSRF)**
  Bezpečností riziko, které naláká přihlášeného uživatele navštívit jinou webovou stránku, kde dojde k vykonávání samotného útoku. Tímto způsobem pak může docházet k mazání či změně dat zejména v administračních prostředích. Jako ochrana před tímto napadením slouží jednoduchý příkaz `$form->addProtection();`. (Nette Foundation, 2016)

Další z uvedených bezpečnostních rizik jsou v Nette automaticky zajištěny. Zde je jejich výčet: *Session hijacking*, *session stealing*, *session fixation*, *URL attack*, *control codes*, *invalid UTF-8*. (Nette Foundation, 2016)

6 Testování

Stejně jako každý softwarový produkt, i daný informační systém přechází z fáze implementace na fázi testování. Je nutné ověřovat správné chování veškerých funkcionalit a podrobit je náhodnému i cílenému zkoušení. Ačkoliv byla v průběhu vývoje vždy průběžně testována drtivá většina součástí systému, mohou se objevovat drobné chyby či překlepy, které mohou rozbít ucelené funkční prvky. Právě z toho důvodu je vhodné provést i testování ve fázi hotového produktu a ověřit si, zda je vše bez problémů použitelné.

6.1 Uživatelské testování

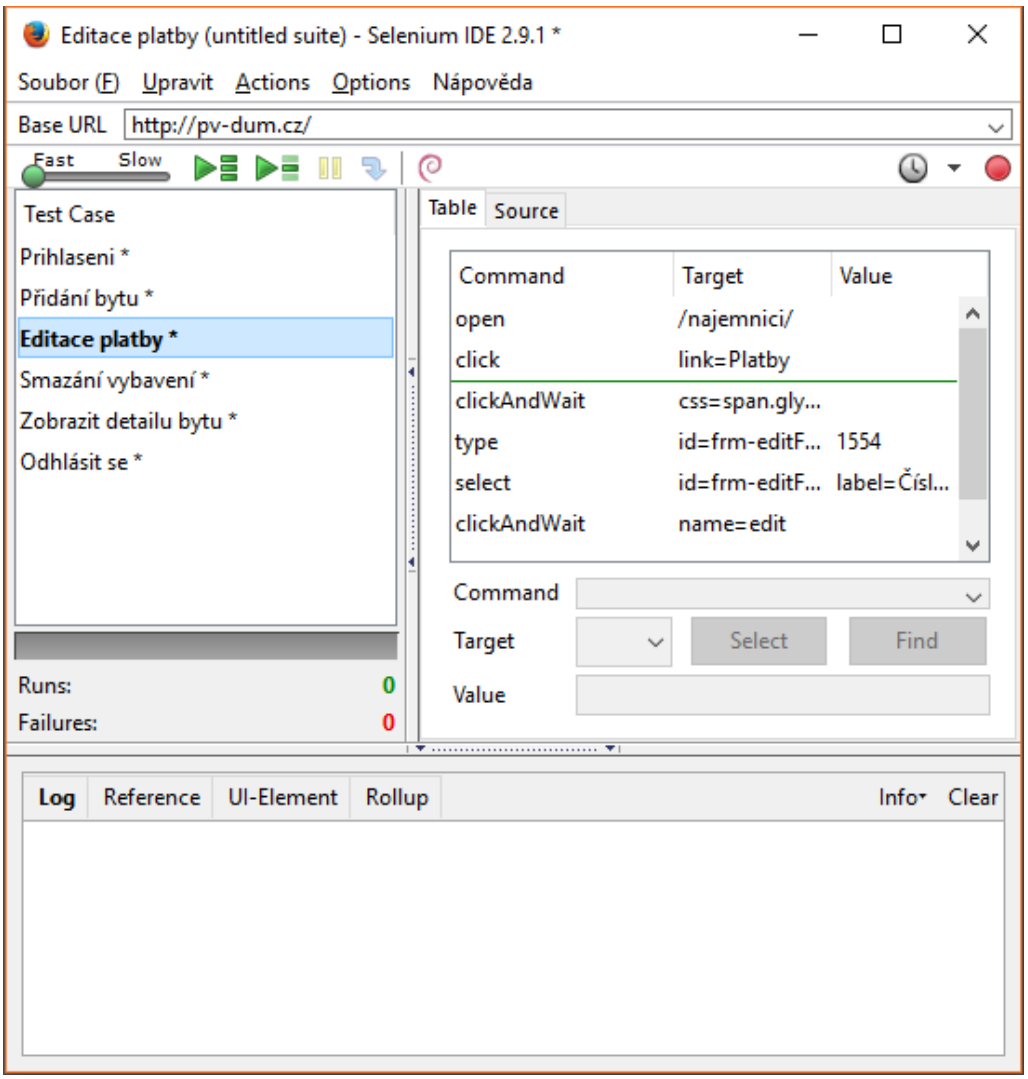
Jednou z forem testování, které je v tomto konkrétním případě velmi důležité, je právě otestování zadavatelem, pro kterého je informační systém vytvořen.

Nejdříve bylo zadavateli předloženo přihlašování, se kterým nebyl nejmenší problém. Poté si pro ukázku vyzkoušel přidat některé logické celky, které potřebuje běžně evidovat. Prvním krokem bylo vytvoření nového záznamu o nájemníkovi. Následovalo přidání nového bytu s jeho veškerými základními údaji. V tomto formuláři také došlo k přiřazení předtím definovaného nájemníka. Poté bylo vyzkoušeno přidání platby, spotřeby energie a vybavení z prostředí detailu bytu.

Tato základní sada evidenčních operací nezaznamenala žádné potíže. Zadavatel vždy věděl přesně, co je jeho cílem a jeho reakce byly velmi rychlé. Poznamenal, že mu velmi vyhovuje uživatelské prostředí a minimum rušivých elementů. Uživatelské testování tedy proběhlo velmi dobře a naplnilo očekávání zadavatele zejména z hlediska jednoduchosti práce s evidovanými údaji.

6.2 Selenium IDE

Selenium IDE je integrované vývojové prostředí, které podporuje Selenium skripty. Je distribuován jako doplněk do internetového prohlížeče Mozilla FireFox. Mezi jeho hlavní funkce patří zejména nahrávání a zpětného přehrávání úkonů na testované webové stránce. Je možné si pro každou logickou operaci vytvořit vlastní test, který je možno dále ladit pomocí krokování a dle potřeb také exportovat do formátu HTML či Ruby skritpu. (Selenium, 2016)



Obrázek 17: Ukázka testů v Selenium IDE

7 Nasazení

Hotový informační systém je v závěru svého vývoje nutné nasadit, aby byl plně přístupný pro zadavatele i jeho další uživatele (nájemníky). V průběhu vývoje bylo využíváno zejména lokálního serveru, který běžel na vývojářském počítači. Z toho důvodu nebylo nutné ihned produkt kamkoli nasazovat. Postupem vývoje však vznikla potřeba vzdálené prezentace rozpracovaných části informačního systému.

7.1 IBM Bluemix

IBM Bluemix je cloudová služba, která umožňuje vývojářům vytvářet rychle a snadno nové projekty. Je založena na *Cloud Foundry* a zajišťuje mnoho funkcí, které umožní jednoduše udržovat či rozšiřovat jakékoli projekty i v týmovém vývoji. Nahrávání samotných souborů je poté řešeno dvěma způsoby. První z nich je propojení s GIT či pomocí CF příkazového terminálu. Prostředí Bluemixu nabízí základní rozcestník (dashboard), kde je možné pomocí pár kliknutí a výběru technologie založit nový projekt. Podpora jazyků je velmi bohatá, je možné zvolit PHP, Python, Ruby, JavaScript, Java, Node.js a desítky dalších variant umístěných v katalogu. Po výběru konkrétní technologie a názvu projektu, následuje možnost připojit přídavné služby. V tomto konkrétním případě bylo například využito databáze. (IBM, 2016)

V průběhu vývoje byl krátce umístěn informační systém na tomto cloudovém řešení. Nicméně v dalších fázích od něj bylo upuštěno zejména z těchto důvodu:

1. Omezená měsíční trial verze
2. Poměrně časté výpadky a údržby serveru poskytující danou službu
3. Složitější správa databáze

7.2 Výsledné produkční prostředí

Po konzultaci se zadavatelem došlo k rozhodnutí nákupu vlastního doménového jména a k tomu i webhostingové služby. Bylo tak rozhodnuto zejména z důvodu nezávislosti a bezpečnosti, které právě tato varianta představuje. Dále zadavatel projevil zájem o možnost sdílet prostřednictvím vlastního webového prostoru i jiné soubory, které nemusí přímo souviset se samotným informačním systémem. Výsledný informační systém tedy běží na doméně *http://www.pv-dum.cz/*, kde je možné si jej prohlédnout v jeho ostré verzi.

8 Závěr

Realizovaný informační systém splnil veškeré požadavky, které na něj byly od počátku vývoje kladeny. Zadavatel projektu přijal výslednou práci velice kladně a souhlasil s logickým uspořádáním veškerých funkcí. Ačkoliv je systém postaven na základě potřeb konkrétní osoby, i přesto je samozřejmě možné jej používat pro kterýkoliv jiný nájemní dům.

Pokud bude pronajímatel během používání systému opravdu poctivě sbírat všechny údaje, bude za to odměněn velmi přesným a uceleným pohledem na to, jak se v nájemním domě hospodaří s penězi, který byt jej stál nejvíce investic, případně zjišťovat, kdy v daném bytě bydlel určitý nájemník a dohledávat i v historických datech jeho osobní údaje.

Jedním z problémů, který pravděpodobně vznikne během užívání systému, je přístup samotných nájemníků. Ačkoliv mají všichni z nich dostupné připojení k Internetu, někteří z nich budou pravděpodobně stále preferovat osobní setkání či domluvu. To ovšem není z hlediska záměru používání příliš velký problém, protože se od začátku počítalo s tím, že komunikace s pronajímatelem prostřednictvím informačního systému bude jen jednou z variant a bude samozřejmě záležet na každém z nájemníků, čemu dá přednost.

Do budoucna by bylo možné daný systém dále rozšiřovat o další evidované údaje, funkcionality. Jednou z rozšíření může být správa více nájemních domů současně, což by vyžádalo rozdělení logiky systémů do více samostatných celků na základě každého domu. Velmi přínosná by byla určitá miniaplikace, která by na základě zadaných údajů ze systému dokázala vytvářet každý rok vyúčtování vodného a stočného a rovnou generovala tisknutelný vzorový dokument.

Dále by bylo možné vytvořit webový portál pro všechny pronajímatele, kde by mohli mezi sebou komunikovat a vyměňovat si tipy a především budovat databázi nájemníků. To znamená, že by poté například mohl jeden pronajímatel varovat druhého tím, že by napsal referenci na konkrétního nájemníka a tím předešel určitým problémům. Dalším zajímavým nápadem bylo změnit informační systém na službu, která by nabízela komukoliv se zaregistrovat a vytvořit pod svým účtem vlastní instanci domu, který by mohl prostřednictvím Internetu spravovat.

9 Seznam zdrojů

Apachefriends. *About the XAMPP project* [online]. [cit. 2016-04-10]. Dostupné z: https://www.apachefriends.org/about.html.

Bootstrap. *Bootstrap · The world's most popular mobile-first and responsive front-end framework.* [online]. [cit. 2016-04-10]. Dostupné z: http://getbootstrap.com/.

Finance media a.s.. *Regulované nájemné - Finance.cz* [online]. [cit. 2016-04-10]. Dostupné z: http://www.finance.cz/bydleni/najemni-bydleni/regulovane-najemne/.

Geekswithblogs. *MVVM Compared To MVC and MVP* [online]. [cit. 2016-04-10]. Dostupné z: http://geekswithblogs.net/dlussier/archive/2009/11/21/136454.aspx.

GILMORE, W. *Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály.* Nové, 3. vyd. Brno: Zoner Press, 2011, 736 s. Encyklopedie Zoner Press. ISBN 978-80-7413-163-9.

Google. *Charts | Google Developers* [online]. [cit. 2016-04-10]. Dostupné z: https://developers.google.com/chart/.

HTML5 UP. *Striped | HTML5 UP* [online]. [cit. 2016-04-10]. Dostupné z: http://html5up.net/stripeds.

IBM. *What is IBM Bluemix?* [online]. [cit. 2016-04-10]. Dostupné z: https://www.ibm.com/developerworks/cloud/library/cl-bluemixfoundry/.

IC SOFTWARE. *Program pro správu budov, domů a nemovitostí* [online]. [cit. 2016-04-10]. Dostupné z: http://www.icsoftware.cz/.

ITnetwork. *MVC architektura* [online]. [cit. 2016-04-10]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/mvc-architektura-navrhovy-vzor/.

JANDA, Pavel. *Datagrid | Ublaboo* [online]. [cit. 2016-04-10]. Dostupné z: http://ublaboo.paveljanda.com/datagrid/.

KOVAŘÍK, Jan. *Sharp and clean symbols - GLYPHICONS.com* [online]. [cit. 2016-04-10]. Dostupné z: http://glyphicons.com/.

MySQL. *MySQL :: Developer Zone* [online]. [cit. 2016-04-10]. Dostupné z: http://dev.mysql.com/.

NetBeans. *NetBeans IDE - Overview* [online]. [cit. 2016-04-10]. Dostupné z: https://netbeans.org/features/index.html.

Nette Foundation. *Zabezpečení před zranitelnostmi* [online]. [cit. 2016-04-10]. Dostupné z: https://doc.nette.org/cs/2.3/vulnerability-protection.

NOVELL. *Benefits of MySQL* [online]. [cit. 2016-04-10]. Dostupné z: http://www.novell.com/documentation/nw65/web$_m$ysql$_n$w/data/aj5bj52.html.

O.K. SOFT. *WinDomy* [online]. [cit. 2016-04-10]. Dostupné z: http://www.oksoft.cz/win-domy/.

PHPMYADMIN. *phpMyAdmin* [online]. [cit. 2016-04-10]. Dostupné z: https://www.phpmyadmin.net/.

RONOVSKÁ, KATEŘINA, LENKA DOBEŠOVÁ A MILOSLAV HRDLIČKA *Jak správně pronajmout, prodat, koupit dům či byt* 1. vyd. Praha: Grada Publishing, 2012, 128 s. Profi hobby, 153. ISBN 978-80-247-4204-5.

SELENIUM. *Selenium IDE Plugins* [online]. [cit. 2016-04-10]. Dostupné z: http://www.seleniumhq.org/projects/ide/.

SYSAG. *Software pro správu nemovitostí* [online]. [cit. 2016-04-10]. Dostupné z: http://www.sysag.cz/.

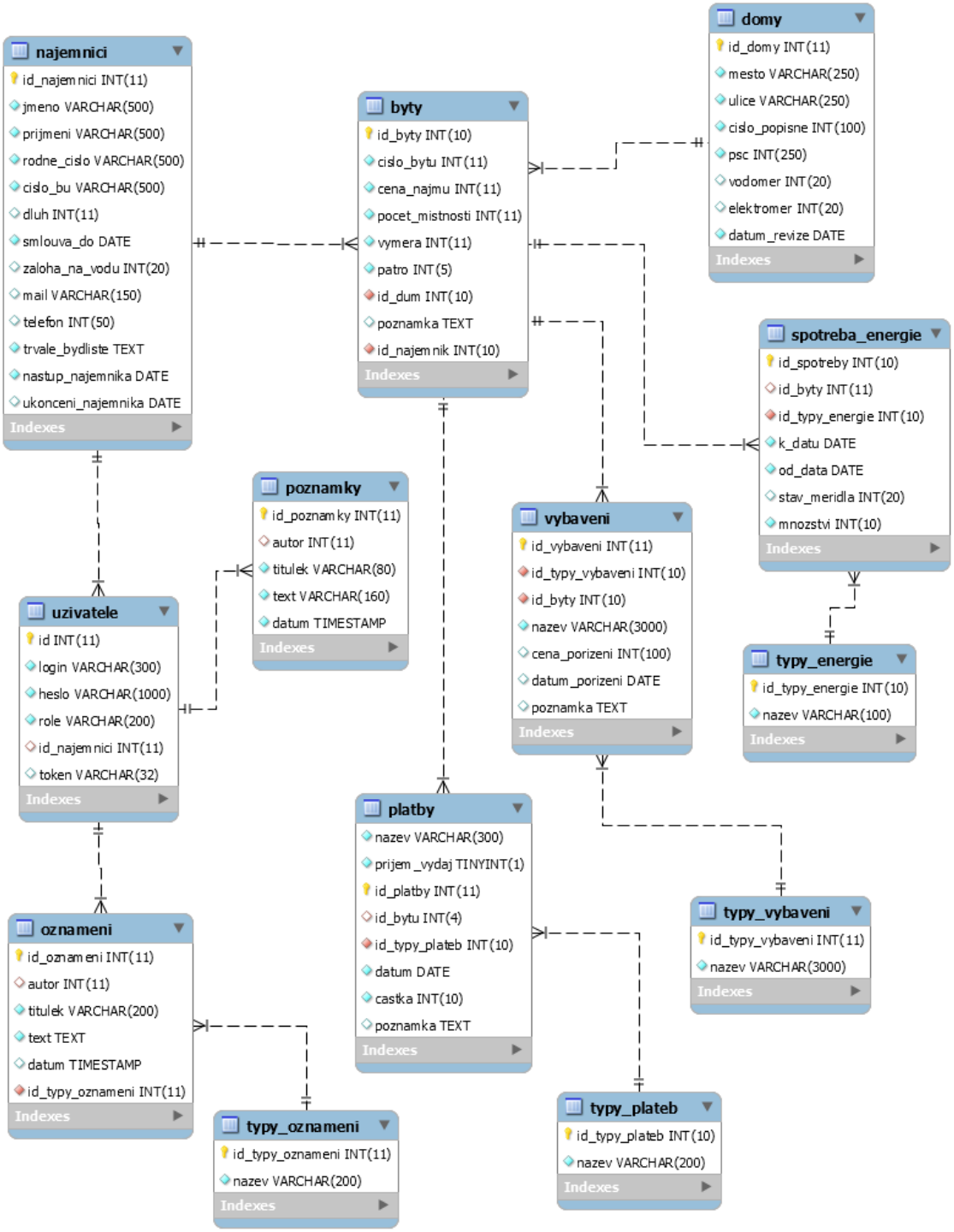
ŠILHAVÝ, RADEK *Vybrané aspekty návrhu webových informačních systémů* Vsetín: Šilhavý, 2013, 1 online zdroj ([12], 164 s.) ISBN 978-80-904741-3-0.

VYDAS. *Správa bytů a nemovitostí* [online]. [cit. 2016-04-10]. Dostupné z: http://www.vydas.cz/sbn/.

ZDROJAK. *Anketa Zdrojáku - Formuláře Google* [online]. [cit. 2016-04-10]. Dostupné z: https://www.zdrojak.cz/clanky/vysledky-ankety-mezi-ctenari-vede-klasika-xhtml-php-a-mysql/.

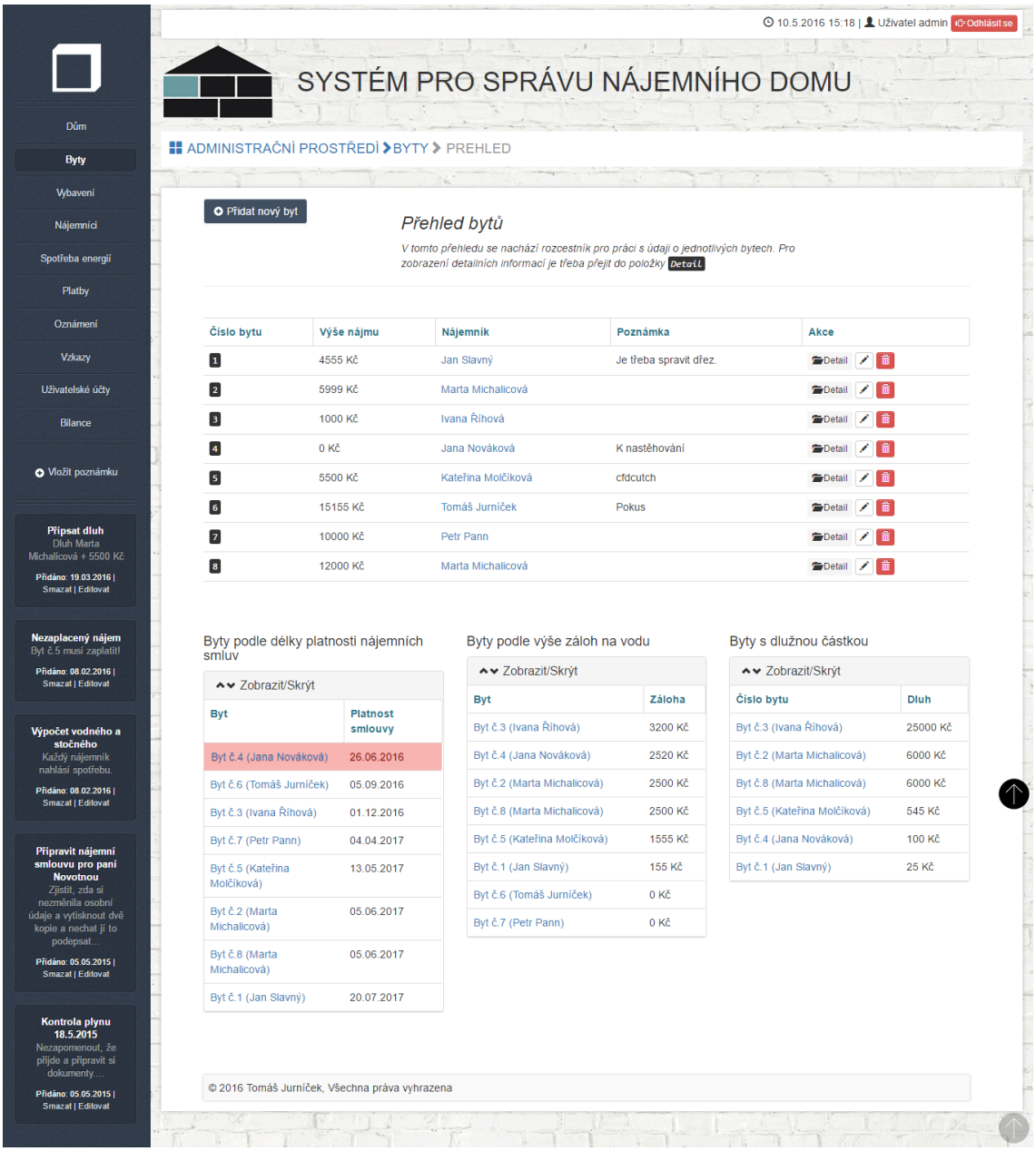
Přílohy

A Fyzický model



Obrázek 18: Fyzický model

B Ukázka systému - přehled bytů



Obrázek 19: Přehled bytů

C Ukázka systému - responzivní pohled

Obrázek 20: Responzivní pohled