# Programátorská dokumentácia

Aplikácia je naprogramovaná v jazyku Ruby On Rails (RoR). Keďže Ruby On Rails je MVC architektúra, jednotlivé časti aplikácie sú tematicky rozdelené do controllórov a modelov. Dáta aplikácie sú uložené v databáze.

Vstupné údaje aplikácia prijíma od užívateľa z jednotlivých rozhraní vo forme textových reťazcov. Dáta sú následne uložené do databáze a v prípade potreby o ne aplikácia požiada prostredníctvom modelov. Modely majú dáta uložené v jednotlivých triedach, v ktorých sú už pretypovaná na základe typu, v akom sú uložené v databáze.

Keďže sa jedná o webovú aplikáciu jediné čo potrebuje k fungovaniu je webový server a databáza. Zvolená architektura RoR podporuje niekoľko webových serverov. Odporúčané sú Apache alebo lighttpd. Ako databázu je možné použiť MySQL, PostgreSQL, SQLite, Oracle, SQL Server, DB2 alebo Firebird. Vďaka tomu je možné spravovať a rozširovať aplikáciu jak pod Windows, tak pod Linuxom.

## Databáza

Dáta celej aplikácie sa ukladajú do MySQL databáze (možné použiť inú). Všetky tabuľky v databáze sú medzi sebou prepojené, takže sa nemôže stať, že v databáze nájdeme údaje, ktoré s ostatnými nesúvisia. Nehnuteľnosti a nájomníci sú uložení v samostatných tabuľkách a navzájom spojené. Sú v nich uložené základné a osobné údaje. Vybavenie nehnuteľnosti, ako aj ostatné údaje (financie, správy, súvisiace s nimi a nájomníkmi) sa nachádzajú vo vlastných tabuľkách, ktoré sú potom k nim pripojené.

Jednotlivé tabuľky majú anglické názvy uvedené v pluráli. Túto konvenciu zavádza architektúra Rails, kvôli lepšej práci s nimi. K jednotlivým údajom v tabuľkách sa potom dostanem cez Model v aplikácii, ktorý má názov v singulári. Ak teda napr. máme tabuľku nájomníkov, jej názov v databáze bude Renters a pristupovať k údajom budeme pomocou Modelu s názvom Renter. Na ukladanie dát do databáze bola zvolená databáza MySQL z toho dôvodu, že je dobre známa a väčšina programátorov s ňou má skúsenosti. Takže pri prípadnom rozširovaní aplikácie by s touto databázou nemali byť problémy.

## Modely

Jednotlivé modely reprezentujú údaje v databáze. Na reprezentovanie dát z databázy Railsy využívajú ORM - object-relational mapping. ORM knižnice mapujú tabuľky databázy do tried. Ak napr. máme v databáze tabuľku s názvom Renters (nájomníci), v programe budeme mať triedu Renter (nájomník). Jednotlivé riadky z danej tabuľky odpovedajú objektom v programe. Jeden konkrétny nájomník je reprezentovaný ako objekt triedy Renter. V rámci objektu sa používajú atribúty ako get a set metódy, ktoré aktualizujú špecifické bunky v tabuľke. V našom prípade trieda Renter má get a set metódy na nastavenie mena, priezviska, adresy atď. (Samotné Railsy pridávajú do tried metódy na prácu s tabuľkou, akými sú napr. vyhľadanie a uloženie záznamu Renter.find, Renter.save). Rails Teda využíva ORM na mapovanie tabuliek do tried, riadkov do objektov a stĺpcov do atribútov.

Na podporu ORM slúži v Railsoch trieda ActiveRecord. Táto trieda zbavuje programátora starostí s pripojením na databázu, programátor sa prostredníctvom tejto triedy môže plne venovať práci s dátami ako takými. ActiveRecord sa ďalej pohodlne a hladko integruje do zbytku Rails architektúry a poskytuje sofistikovanú validáciu dát, ktoré dokáže extrahovať z webových formulárov a uložiť do modelu. Trieda ActiveRecord dokáže zabezpečiť extrakciu a formátovanie chýb pomocou jedného riadku s kódom. Táto trieda predstavuje dôležitý základ celej Rails MVC architektúry.

Databáza obsahuje 19 tabuliek s údajmi. Jednotlivé modely aplikácie sú:

* **Accesorry –** reprezentuje príslušenstvo nehnuteľnosti, ktoré tvorí jeho stálu súčasť a je účtované jeho používanie. Trieda Accessory viazaná na nehnuteľnosť a v aplikácii sa jej údaje využíva na generovanie evidenčného listu.
* **AddedPenalty -** prostredníctvom tejto triedy je v aplikácii definovaný vzťah n..n medzi triedami Bill a Penalty. Trieda generuje počet penále, ktorý má byť pripočítaný k platbe, ktorej vypršala doba splatnosti a históriu pripočítania penále. Z údajov o penále zistí, po koľkých dňoch sa penále aplikuje a v závislosti na aktuálnom dni vypočíta, koľkokrát sa má dané penále aplikovať a údaj o počte uloží do databáze. Penále je potom zobrazené pri danej platby v sekcii financie. Výsledné penále ma prenajímateľ možnosť upraviť a po zaplatení, sa výška penále uloží do tabuľky AddedPenalties, a trieda AddedPenalty vygeneruje históriu pridelenia jednotlivých penále, aby mal prenajíteľ prehľad o pripočítaných penále.
* **Additional –** trieda reprezentuje údaje nehnuteľnosti a nájomníka, ktoré má k nim prenajímateľ možnosť definovať naviac. Dáta sú uložené v jednej tabuľke, pretože majú rovnaké atribúty. Jediný rozdiel je ich príslušnosť k nehnuteľnosti alebo nájomníkovi. Tú reprezentuje výčtový typ ENUM v tabuľke, podľa ktorého aplikácia zistí, komu údaj patrí.
* **AutoPay –** automatické platby, ktoré si zadefinuje prenajímateľ viazané na nehnuteľnosť alebo nájomníka reprezentuje táto trieda. V tabuľke AutoPays sú uložené jednotlivé definície automatických platieb a z ich údajov sa vygenerujú platby. Na základe údaju o počte dní opakovania a dátumu posledného generovania vygeneruje platby do konca aktuálneho mesiaca. Platby su reprezentované triedou Bill.
* **Bill –** jednotlivé platby v aplikácii reprezentuje táto trieda. Pomocou nej získava prenajímateľ informácie o platbách, umožňuje mu evidovať platby ako zaplatené v sekcii financie. Platbu môže evidovať ako čiastočne zaplatenú, alebo plne zaplatenú. Pri čiastočnej platbe sa vytvorí nová platba, ktorá má poznámku, že je čiastočná a cena pôvodnej platby je znížená. Pôvodnej platbe je taktiež pridelená poznámka, že bola čiastočne zaplatená. Trieda Bill je ďalej zodpovedná za výpočet penále, na ktorom spolupracuje s triedou AddedPenalty. Tá poskytuje výpočet počtu penále a trieda Bill ďalej na základe tohto počtu toto penále pripočíta k danej platbe. Trieda Bill získava u databáze informácie o platbách a na základe nich vypočítava príjmy a výdaje. S týmito údajmi ďalej pracuje controller štatistiky. Controlleru trieda poskytuje dáta na zobrazovanie grafu v sekcii štatistika.
* **Contact -** trieda reprezentujúca kontakty prenajímateľa. Aplikácia ju využíva na vytváranie a ukládanie kontaktov k jednotlivým nehnuteľnostiam.
* **Fitting –** slúži na reprezentáciu vybavenia jednotlivých nehnuteľností.
* **MyDate -** pomocná tried, ktorá nie je viazaná na tabuľku v databáze. Táto trieda slúži na formátovanie časových údajov a dátumov. Časové údaje, ktoré sa ukladajú do databázy, ako napríklad deadline splatnosti nejakej platby, sú ukladané v sekundách. Pomocou tejto triedy sa uložené sekundy prevedú na dátum a čas.
* **Note –** umožňuje priradiť poznámky k platbám a nehnuteľnostiam.
* **PenaltyHistory –** ukladá do databázy informácie o pripočítanom penále k jednotlivým platbám. Každý záznam obsahuje údaj o pôvodnej výške a novej pripočítanej výške penále.
* **Penalty –** reprezentuje jednotlivé penále, ktoré užívateľ
* **Property –** slúži na prácu s nehnuteľnosťou. Táto trieda sa v aplikácie využíva na overenie príslušnosti získaných údajov prihlásenému užívateľovi. Keďže takmer všetky dáta z tabuliek v databáze, sú prepojené s touto triedou, k tabuľke Properties (nehnuteľnosti) je pridaný atribút user\_id, ktorý reprezentuje jej príslušnosť k užívateľovi. Preto predtým, než sa akékoľvek dáta zobrazia užívateľovi, sú najprv testované, či skutočne patria prihlásenému užívateľovi. Táto ochrana sa využíva v prípade, že by sa prihlásený užívateľ snažil získať dáta, ktoré mu nepatria modifikovaním zdrojových kódov HTML stránky.
* **Rent –** trieda reprezentujúca jednotlivé nájomné nájomníkov. Využíva sa na uloženie informácii o zmenách nájomného v priebehu ubytovania. Každý nájomník má svoje nájomné uložené v tabuľke Rents a pri zvýšení, prípadne znížení sa vytvorí nový záznam. Staré nájomné v tabuľke ostáva spolu s novým, tým pádom má užívateľ k dispozícii históriu zmeny nájomného.
* **Renter –** trieda reprezentujúca nájomníka nehnuteľnosti. Okrem získavania základných údajov nájomníka z databáze sa využíva na výpočet príjmov a výdajov daného nájomníka a výpočet dlžnej čiastky.
* **Room –** reprezentuje dáta z tabuľky Rooms, kde sú uložené názvy základných izieb v nehnuteľnosti akými sú kuchyňa, kúpeľňa, balkón, . K jednotlivým nehnuteľnostiam sú potom tieto izby referencované vzťahom n..n. Každá izba má údaje o svojej rozlohe. Tieto údaje má na starosti trieda RoomArea.
* **RoomArea –** táto trieda je použitá na prepojenie tabuliek Properties a Rooms, ktoré majú medzi sebou vzťah n..n. K jednotlivým izbách pridáva informácie o ich rozlohách.
* **Todo –** trieda reprezentuje úlohy a pripomienky, ktoré sú priradené k jednotlivým nehnuteľnostiam.
* **User –** pre každého užívateľa, ktorý si zadefinuje nejaké nehnuteľnosti v aplikácii, je v databáze vytvorený záznam s jedinečným id, ktoré je pridané ku každému záznamu k jeho nehnuteľnostiam. Trieda potom prostredníctvom tohto id a faktu, že všetky údaje (nájomníci, vybavenie nehnuteľnosti, platby, penále atď.) sú viazané na nejakú nehnuteľnosť, získava z databáze iba údaje, ktoré patria prihlásenému užívateľovi. Trieda ďalej poskytuje autorizáciu užívateľa pri prihlásení. Pri vytvorení nového užívateľa jeho heslo zahashuje do databáze algoritmom SHA1 a pri prihlásení overí jednotlivé hashe hesla.

## Controllery aplikácie

### Controller financií

Všetky operácie, ktoré súvisia s financiami a jej správou má na starosti tento controller. Patrí sem správa platieb, automatických platieb a penále k platbám. Jeho hlavnou funkciou je vygenerovať platby na aktuálny mesiac podľa predpisu automatických platieb, spracovávať platenie jednotlivých platieb a výpočet penále za oneskorené platby.

Generovanie automatických platieb spustí controller prostredníctvom triedy AutoPay. Tá reprezentuje automatickú platbu a jednotlivé automatické platby generuje vždy na aktuálny mesiac podľa počtu dní opakovania danej automatickej platby. Po tejto generácii už má aplikácia k dispozícii platby, z ktorých môže vypočítať očakávaný príjem, informácie o ktorom ma prenajímateľ v sekcii štatistika.

Výpočet penále prebieha prostredníctvom triedy AddedPenalty, cez ktorú je v aplikácii definovaný vzťah n..n medzi triedami Bill a Penalty. Táto trieda zároveň slúži na generovanie histórii penále, z ktorých sa prenajímateľ dozvie v ktorý deň aká hodnota penále bola pripočítaná. Celkové pridané penále má však prenajímateľ možnosť pred započítaním zmeniť.

Controller ďalej zobrazuje užívateľovi jednotlivé platby. Triedou User získa všetky platby z databáze, ktorá ich vyhľadá v prostredníctvom id prihláseného užívateľa, ktoré je uložene v session pri prihlásení. Prostredníctvom triedy Bill im ich umožní užívateľovi spravovať (zaplatiť, zmeniť).

### Správa nehnuteľností

Controller v tejto sekcii poskytuje prenajímateľovi prístup k správe svojich evidovaných nehnuteľností. Základ všetkých controllorov sú funkcie na podporu pridávania, úpravy a mazania údajov v databáze. Okrem správy samotnej nehnuteľnosti controller umožňuje vytvárať a pridávať poznámky a vybavenie k jednotlivým nehnuteľnostiam.

Keďže sa jedná o nehnuteľnosti, controller úzko spolupracuje s triedou Property, ktorá reprezentuje nehnuteľnosť a obsahuje dáta z databázy. Hlavnou úlohou controlleru pre nehnuteľnosti je poskytnúť prenajímateľovi informácie o nej, ktoré môže upravovať a následne tieto informácie poskytne nájomníkom, ktorí k nim budú mať prístup z ich zohrania.

Ako je už spomenuté v úvode, sekcia správy nehnuteľnosti ma niekoľko podkategórii. Tieto podkategórie predstavujú v controlleri jednotlivé akcie (akcia = metóda controlleru), ktoré controller zavolá na základe odozvy užívateľa, ktorý s aplikáciou pracuje. Každá akcia ma svoj pohľad, ktorý poskytne prenajímateľovi informácie, ktoré potrebuje.

Controller nehnuteľností, podobne ako controller správy nájomníkov, spolupracuje aj s inými modelmi než len s nehnuteľnosťami. V podkategórii financie sa nachádza správa financií, kde má prenajímateľ prehľad a správu všetkých financií, ktoré súvisia s danou nehnuteľnosťou. Údaje o financiách dostáva prostredníctvom triedy Bill, ktorý reprezentuje jednotlivé platby. Platby môže prenajímateľ zaplatiť a v tejto kategórii sa k tomu využíva technológia AJAX (Asynchrónny Javascript a XML. Technológia slúži na prácu s dátami na stránke bez nutnosti jej znovunačítania). Aby sa nemusela cela stránka s platbami obnovovať, AJAX obnoví iba určité elementy na stránke, čím má prenajímateľ väčší prehľad, čo sa s jeho platbami deje. Akcia, ktoré je zavolaná na zaplatenie platieb zistí, či bola zavolaná ako XMLHTTPrequest a podľa toho obnoví buď celú stránku, alebo iba jej časť, ktorej identifikáciu získa zo spomínaného requestu.

Zo sekcie nehnuteľností je pre zadávanie ďalších prídavných informácii venovaná podkategória vybavenie. Tu je možnosť pridať nehnuteľnosti jej špecifické informácie od vybavenia jednotlivých miestností až po služby spojené s nehnuteľnosťou. Dáta z tejto kategórie sú taktiež reprezentované samostatnými triedami a ich hodnoty sú neskôr použité na generovanie evidenčného listu. V tejto podkategórii controller využíva triedy Accessory, Fitting a Contact.

Controller nehnuteľnosti má ďalej za úlohu vygenerovať evidenčný list. Jeho obsah získa prostredníctvom triedy Accessory, RoomArea a Property.

### Správa nájomníkov

V tomto controlleri sú podobné akcie na správu nájomníkov, ako v controlleri pre nehnuteľnosti. Dôležité funkcie na správu nájomníkov poskytuje samotný controller rovnako ako akcie na prechod k jednotlivým podkategóriám. Controller komunikuje hlavne s triedou Renter.

Obidve sekcie nehnuteľností a nájomníkov sú zároveň miesta, kde je prenajímateľ upozornení na prípadne chýbajúce údaje a na udalosti, spojené s prenájmom. Typicky sa tento kód umiestňuje do pohľadov no v tomto prípade by umiestnenie kódu na generovanie týchto informácií v pohľadoch spôsobilo zavlečenú zložitosť a tým pádom zneprehľadnenie celého kódu. Riešením sú Hepler moduly. Helper moduly sú vygenerované ku každému controlleru. Je to ďalšia šikovná vlastnosť architektúry Rails. Helper moduly sa využívajú vtedy, ak chceme zaviesť určitú abstrakciu nejakých operácii do pohľadov. Do týchto modulov môže programátor umiestniť pomocné formátovacie metódy, ktoré by inak v kóde spôsobovali jeho neprehľadnosť a ktoré funkčne nepatria ani do controllera ani do modelu. Väčšinou sa tieto moduly používajú na úpravu údajov, ktoré sa zobrazujú pohľadoch resp. ich volanie je možné iba z jednotlivých pohľadov. Pre controller a model nie sú viditeľné (a potrebné). Tie v tomto miesto poskytli presne tú funkčnosť, ktorú aplikácia potrebovala. Miesto desiatok riadkov kódu v pohľade bol tento kód (na generovanie varovaní) umiestnený do Helperov a v pohľade sa jednoducho zavolá funkcia, ktorá jednotlivé varovania vygeneruje.

Správa nájomníkov poskytuje aj informácie o jednotlivých platbách, ktoré získava od triedy Bill, podobne ako je tomu u finančného controlleru. Platby sa ale týkajú iba konkrétneho nájomníka, podobne to je aj v sekcii s nehnuteľnosťami, ktoré taktiež poskytujú informácie o platbách ale iba v rámci danej nehnuteľnosti. Zobrazenie platieb , ktoré súvisia iba s nájomníkom zabezpečí controller financií, ktorý to zistí z parametrov pri prechode na podkategóriu financie v správe nájomníkov.

### Controller štatistiky

Okrem zoznamu financií bolo nutné, aby mal užívateľ tiež nejaký celkový finančný prehľad, aký bol príjem jednotlivých nehnuteľností, ich výdavky a podobne. V prípade takýchto prehľadov sú dobrým nástrojom grafy, ktoré umožňujú užívateľovi vytvoriť si lepšiu predstavu o finančnom (a inom) stave.

Na vykreslenie grafu v aplikácii, ktorý by zobrazil výšku príjmu v závislosti na dni v mesiaci sa použil plugin do RoR (http://www.pullmonkey.com/projects/open\_flash\_chart), ktorý je voľne dostupný a je poskytuje niekoľko typov prehľadných grafov, ktoré sú naviac interaktívne. Grafu stačí dodať potrebné informácie o platbách (naplniť pole) a nastaviť názov jednotlivých ôs. Užívateľ má potom pred sebou zrozumiteľný prehľad svojich príjmov. Výhodou grafu je, že programátor nemusí nič programovať, stačí mu predať dáta a nastaviť názvy pre jednotlivé osi.

Graf je implementovaný pomocou jazyka ActiveX a Flashu. Práve vďaka Flashu pôsobí na užívateľa príjemným dojmom. Aby sa pohodlie práce s grafom ešte zvýšilo, aplikácia opäť využíva technológiu AJAX, na jeho zobrazenie a update údajov. Graf je interaktívny a pri presune kurzoru myši nad jednotlivé príjmy sa zobrazia informácie o platbách v daný deň.

### Controller pripomienok

Dôležitou funkciou, ktorú mala aplikácia poskytovať bolo tiež schopnosť upozorniť prenajímateľa na blížiacu sa udalosť, ktorú si uložil k nehnuteľnostiam, alebo priamo do aplikácie bez akejkoľvek viazanosti na nehnuteľnosť.

Správu jednotlivých udalostí ma za úlohu controller pripomienok, v aplikácii zobrazovaný ako Úkolovník. Prostredníctvom neho umožňuje aplikácia prenajímateľovi poznamenať si k jednotlivým nehnuteľnostiam rôzne úlohy, ktoré treba splniť. Controller potom na základe voľby uloží pripomienky do databáze (prostredníctvom triedy Todo) s príslušným stupňom dôležitosti a príznakom, kde ma aplikácia na danú udalosť prenajímateľa upozorniť (po prihlásení, v detaile nehnuteľnosti, prípadne neupozorňovať).

### Controller kontaktov

Prostredníctvom tohto controllera má prenajímateľ možnosť priradiť nehnuteľnostiam kontaktné informácie na dodávateľov (opravár, upratovanie, právnik atď.) Tento controller má význam hlavne pre nájomníkov, pretože ním prenajímateľ poskytne informácie, na koho sa obrátiť v prípade nejakej nehody a pod. Controller spolupracuje s triedou Contact, ktorá má na starostí ukladanie kontaktov do databáze.

### Login controller

Poskytuje správu užívateľských účtov. Prenajímateľ ma prostredníctvom neho možnosť vytvoriť prihlasovacie meno a heslo pre svojich nájomníkov. Login controller má ďalej na starosti správu prihlasovania do aplikácie a po prihlásení zobrazí prenajímateľovi informačnú tabuľku, kde poskytne dôležité údaje akými sú už spomínané pripomienky, upozorní na blížiace sa ukončenie pobytu nájomníka a na dlžníkov.