UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Domáce účtovníctvo

2013

Peter Kotulič

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Domáce účtovníctvo

Bakalárska práca

Študijný program : Aplikovaná informatika
Študijný odbor: 9.2.9 Aplikovaná informatika
Školiteľ: Mgr. Ján Kľuka, PhD.

Bratislava, 2013

Peter Kotulič

Univerzita Komenského v Bratislave
Fakulta Matematiky, Fyziky a Informatiky

Zadanie

Meno a priezvisko študenta: Peter Kotulič

Študijný program: aplikovaná informatika

Typ záverečnej práce: bakalárska

Jazyk záverečnej práce: slovenský

Názov: Domáce účtovníctvo

Cieľ práce: Navrhnúť a implementovať webovú aplikáciu umožňujúcu používateľom udržiavanie prehľadu o príjmoch a výdavkoch ich rodinných rozpočtov v minulosti a jednoduché formy plánovania do budúcnosti. Prispôsobiť najčastejšie potrebné funkcie mobilnému použitiu. Využiť framework pre vývoj webových aplikácií, návrhové vzory a rigorózne testovanie.

Školiteľ: Mgr. Ján Kľuka, PhD.

V Bratislave 22. októbra 2012

doc. RNDr. Mária Markošová, PhD.
gestor št. programu

Peter Kotulič
študent

Mgr. Ján Kľuka, PhD.
vedúci, resp. školiteľ

Abstrakt

KOTULIČ, Peter: Domáce účtovníctvo. [Bakalárska práca] – Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej informatiky. – Školiteľ: Mgr. Ján Kľuka, PhD..

TODO

Kľúčové slová: účtovníctvo, financie, framework, MVC, CakePHP

Abstract

KOTULIČ, Peter: Home accounting software. [Bachelor thesis] – Comenius University in Bratislava. Faculty of Mathematics, Physics and Informatics; Department of Applied Informatics. – Supervisor: Mgr. Ján Kľuka, PhD..

TODO

Keywords: accounting, finance, framework, MVC, CakePHP

Čestne prehlasujem, že som túto bakalársku prácu vypracoval(a) samostatne s použitím citovaných zdrojov.

V Bratislave XX. júna .....

Obsah

Kapitola 1

Úvod

1.1 Motivácia

Pri tvorbe tejto práce ma najviac ovplyvnila aktuálna situácia v našej spoločnosti, ktorá je čím ďalej, tým viac konzumne zameraná. Nakupujeme oveľa viac ako v minulosti. A nejde len o potrebné veci. A čím viac vecí nakúpime, tým menší prehľad v nich máme. A tak nakupujeme ešte viac. Tento konzumný spôsob života vplýva najviac na našu peňaženku. A rovnako ako máme problém vyznať sa vo veciach ktoré nakupujeme, rovnako vzniká problém vyznať sa aj v našich financiách. A toto by som rád vyriešil. Verím tomu, že čím viac budeme mať svoje financie pod kontrolou, tým lepšie sa nám bude žiť. Čím lepšie budeme vedieť na čo míňame, tým ľahšie sa nám bude dať naplánovať na ktorých veciach môžeme ušetriť. Mať prehľad a začať spravovať svoje domáce účtovníctvo je tým prvým krokom, ktorý musíme urobiť.

Ďalšou významnou motiváciou k tvorbe tejto práce bola pre mňa túžba naučiť sa niečo nové. Niečo, čoho znalosti mi veľmi dobre poslúžia aj pri mojich budúcich projektoch. V tomto prípade ide o použitie *frameworku* (vývojového rámca) na tvorbu webových aplikácii. Bude sa na ňom zakladať celá moja aplikácia na domáce účtovníctvo.

1.2 Ciele práce

Hlavným cieľom tejto práce je vytvoriť webovú aplikáciu Domáce účtovníctvo na jednoduchú a prehľadnú správu osobných (domácich?) financií. Keďže je už podobných aplikácii dosť veľa, mojím plánom nebude urobiť len ich jednoduchú kópiu. Budem sa

snažiť porovnať tieto existujúce aplikácie, vyhodnotiť ich klady a zápory, a z každej si zobrať len to najlepšie, čo implementujem do mojej výslednej aplikácie. Zamerať sa chcem taktiež aj na jednoduché a prehľadné grafické rozhranie aplikácie, v ktorom sa používateľ bude vedieť orientovať po pár kliknutiach myšou. Toto je jedna z vecí, ktorú by už existujúci podobné aplikácie mali prepracovať. Pri ich skúmaní sa mi často stávalo, že som nevedel a musel dlho hľadať ako vykonať určitú akciu. A toto sa pri aplikáciách, ktoré sú určené bežným používateľom nesmie stávať. Podobným jednoduchým a prehľadným štýlom bude spravená aj mobilná verzia aplikácie. V jednoduchosti je krása. A tak myslím, že je zbytočné robiť prekomplikovanú aplikáciu s „milión“ funkciami, keď ju používateľ po piatich minútach vypne, lebo sa v nej nevyzná.

Kapitola 2

Východiská

V tejto časti práce sa budem snažiť opísať východiská, ktoré ma inšpirovali a pomohli mi pri tvorbe bakalárskej práce. Okrem opisu základnej teórie potrebnej na realizáciu práce pôjde aj o prehľad funkčných požiadaviek na výslednú aplikáciu.

Taktiež spravím prehľad už implementovaných, podobných riešení na správu domáceho účtovníctva. Budem sa zaoberať tromi hlavnými typmi aplikácií. Pôjde o webové, mobilné a desktopové aplikácie na správu a prehľad financií v domácnosti. Mojou snahou je zhrnúť základné vlastnosti a funkcionalitu týchto aplikácií, z každej si zobrať to najlepšie a implementovať to do mojej práce v čo najväčšej miere.

Ďalšiu časť tejto kapitoly bude tvoriť popis Model–view–controller architektúry(MVC) [1] , ktorá bude s pomocou frameworku CakePHP použitá aj v mojej práci. V súčasnosti ide o jeden z najobľúbenejších spôsobov tvorby aplikácií. Keďže som už spomenul aj framework v ktorom budem robiť aplikáciu, rád by som časť tejto kapitoly venoval aj jemu. Nakoniec by som sa rád pozrel a priblížil aj tému návrhových vzorov, ktoré takáto práca bude vyžadovať.

2.1 Použité webové technológie

2.1.1 HTML a CSS

HTML [19] je skratka pre HyperText Markup Language. Je to najpoužívanejší značkovací jazyk na zápis hypermediálnych dokumentov, ktorý používame pri tvorbe web stránok. Štruktúru HTML dokumentu určuje definícia typu (Document Type Definition; DTD). Táto definícia určuje množinu značiek, ktoré môžu byť použité v dokumente.

Aktuálna verzia tohto jazyka je HTML 5 [12] . Táto verzia priniesla mnoho výhod. Okrem tagu video, ktorý by sa mal stať novým štandardom na prezeranie videí a filmov na webe ide hlavne o nové štrukturálne elementy.

Element `div` teraz môže byť nahradený presnejšie definovanými elementami ako sú `header`, nav, section, article, aside a footer. Na jednej strane to pomáha k prehľadnejšiemu kódu a na druhú stranu aj koncovému používateľovi pri jednoduchšom pohybe na stránke. Napríklad môže ísť o rýchle preskočenie navigácie alebo jednoduchší pohyb medzi rôznymi článkami.

Dalšou novinkou sú nové atribúty ktoré môžeme využiť pri formulároch a inputoch. Pre mňa poslednou dôležitou výhodou HTML 5 ktorú by som rád spomenul je použitie Inline SVG(Scalable Vector Graphics) . Ide o vektorovú grafiku ktorú používa aj mnou neskôr spomínaná javascriptová Highcharts knižnica.

CSS [18] (Cascading Style Sheets - kaskádové štýly) je štýlovací jazyk, ktorý popisuje ako budú zobrazené stránky, ktoré boli napísané v (X)HTML. CSS umožňujú vizuálne formátovať webové dokumenty. Vytvárame tak štruktúrované dokumenty v ktorých oddeľujeme obsah dokumentu(HTML) od jeho vzhľadu(CSS). Hlavné z výhod použitia CSS su prehľadnejší kód, menšia dátová náročnosť a jednoduchá zmena formátovania určitých elementov na celej stránke bez nutnosti formátovania každého elementu zvlášť.

2.1.2 CakePHP

CakePHP [7] je open source webový framework určený na vývoj webových aplikácií v programovacom jazyku PHP verzie 4 a 5. Tento framework vznikol už v roku 2005 ako reakcia na veľmi obľúbený a úspešný framework Ruby on Rails. Framework CakePHP je postavený na softwareovej architektúre Model-view-controller, ktorý bližšie opisujem v kapitole 2.4.

2.1.3 Highcharts plugin

Tento plugin [8] pre framework CakePHP využíva niektoré skvelé možnosti knižnice pre tvorbu grafov Highcharts. Hlavnou výhodou tejto knižnice je, že je kompletne napísaná v JavaScripte. Pre jej fungovanie nie sú potrebné žiadne client side pluginy akými sú napríklad Java alebo Flash. Knižnica Highcharts funguje vo všetkých moderných webových prehliadačoch a nerobia jej problém ani tie mobilné. Ako na Androide, tak aj na iOS. Použitý plugin pre CakePHP je stále vo vývoji, a preto zatiaľ neponúka všetky výhody Highcharts. Základná funkcionalita ktorú poskytuje však plne postačuje na využitie v mojej aplikácii.

2.2 Existujúce riešenia

Pred začatím tvorby svojej aplikácie som urobil prehľad podobných existujúcich riešení. Rozdelil som ich do troch kategórií. Webové aplikácie, desktopové aplikácie a mobilné aplikácie. Na začiatok by som rád uviedol, že všetky ďalej spomínané aplikácie spĺňali určitú základnú funkcionalitu, ktorú od aplikácie na prehľad rozpočtu berieme za samozrejmú.

- pridávanie, editovanie a mazanie jednorazových príjmov a výdavkov
- pridávanie, editovanie a mazanie opakovaných príjmov a výdavkov

2.2.1 Webové aplikácie

Budget Pulse [13]

- jednoduchá administrácia a prehľady
- komplikovanejšie/menej prehľadné pridávanie výdavkov a príjmov
- základná funkcia v aplikácii je pridanie transakcie, v následnom menu si používateľ vyberie či ide o príjem alebo výdavok, určí jeho kategóriu, popis, sumu, množstvo, konto(účet) ktoré chce použiť, dátum, prípadné zadanie opakovania transakcie
- možnosť rozdelenia transakcie na menšie časti a ich priradenie do rôznych kategórií

- možnosť pridania viacerých kont
- dáta import, export
- grafy a prehľady:
  - koláčový graf (kategórie za určité obdobie)
  - stĺpcový graf (kategórie/suma za určité obdobie)
  - tabuľkové prehľady (kategórie/čas)
- možnosť určenia osobných cieľov (používateľ si nastaví sumu, ktorá sa mu napríklad raz za mesiac odčíta z konta a priradí k určitému osobnému cieľu, na ktorý si chce našetriť)

Buxfer [6]

- oveľa prehľadnejšia aplikácia ako budgetpulse
- veľmi dobre spracovaná a prehľadná hlavná stránka
  - zobrazuje stav účtu, výdavky za posledný mesiac koláčovým grafom
  - prehľad mesačného budgetu podľa kategórií
  - pripomienky o budúcich výdavkoch (grafický mesačný kalendár)
  - tabuľkové zobrazenie transakcií (popis, suma, tagy, účet, dátum)
  - prehľad požičaných peňazí
- pridávanie transakcií (príjem, výdavok, transfer, refund), upload výpisov z banky
- grafy a prehľady:
  - koláčový graf (kategórie za určité obdobie)
  - stĺpcový graf (čas(x)/suma(y))
  - tabuľkové prehľady (popis, suma, tagy, účet, dátum), možnosť zatriedenia podľa dátumov alebo sumy
- vypočítavanie prehľadov do budúcna na základe predchádzajúcich príjmov a výdavkov

2.2.2 Desktopové aplikácie

RQ Money [17]

komplexný správca financií od slovenského autora vytvorený v Delphi

- podrobná evidencia transakcií (príjmov, výdavkov a presunov)
- filtrovanie, radenie a sumarizácia záznamov podľa vlastného výberu
- plánovanie transakcií s možnosťou prezerania platobného kalendára
- prehľadné tlačové zostavy, exporty tabuliek (napr. pre MS Excel) a grafov
- porovnanie plánovaných a skutočných transakcií za mesiac
- v štatistike podrobné prehľady za rôzne časové obdobia (tabuľky i grafy)
- možnosť použiť vlastné SQL príkazy
- obsahuje viacjazyčnú podporu
- každá transakcia je spojená s účtom, osobou a kategóriou

2.2.3 Mobilné aplikácie

Zaoberal som sa len aplikáciami pre systém Android

Easy Money - Android [10]

- rôzne typy účtov (napríklad cash, kreditná karta, účet v banke,...)
- pridávanie transakcií (názov, cena, kategória, dátum, príjem alebo výdavok, nastavenie opakovania, možnosť rozdelenia transakcií do viacerých kategórií, prevod medzi účtami)
- prehľady:
  - príjmy alebo výdavky podľa kategórií (stĺpcové grafy)
  - mesačný zoznam príjmov a výdavkov (mesiac(x)/suma(y))
  - denné porovnanie príjmov a výdavkov

- pripomienkovač platenia faktúr/účtov
- nastavenie mesačného rozpočtu a prehľad jeho využitia počas mesiaca
- import a export do QIF a CSV
- prehľadávanie medzi transakciami pomocou keywords, triedenie vyhľadávania podľa dátumu, sumy alebo názvu

Money Lover - Android [4]

- najprehľadnejšia a najintuitívnejšia mobilná aplikácia, ktorú som testoval
- pridávanie transakcií:
  - príjem, výdaj, pôžička alebo dlh
  - názov, kategória, suma, poznámka, dátum, účet/konto, nastavenie opakovania (nedá sa nastaviť konkrétny dátum, ale len začiatok alebo koniec týždňa, mesiaca...)
- možnosť prehľadu transakcií, kategórií alebo podľa príjmu a výdavkov
- rôzne typy účtov(kont)
- správa dlhov
- plánovanie rozpočtu (plánovanie šetrenia, pridanie rozpočtu na určité časové obdobie – upozornenie pri dosiahnutí určitej hranice)
- sprievodca pri nastavení aplikácie
- grafy a prehľady:
  - koláčový graf (kategórie za určité obdobie)
  - príjmy verzus výdavky za vybrané obdobie (dátum (x) / suma (y)
  - tabuľkové prehľady (popis, suma, tagy, účet, dátum), možnosť zatriedenia podľa dátumov alebo sumy
  - tabuľkové mesačné výpisy (príjem, výdaj, dlh, pôžička)
- ďalšie možnosti:

  - prepočet meny
  - výpočet vrcholu
  - vyhľadanie bankomatu
  - vyhľadanie banky
  - počítanie úroku

Expense Manager - Android [3]

- pridávanie príjmov a výdavkov
  - účet, dátum, suma, názov, popis, číslo účtenky (aj v ostatných aplikáciách)
  - možnosť nastavenia predvypĺňaných polí
  - pridávanie opakujúcich sa príjmov a výdavkov
- pridávanie rozpočtu na opakujúce sa dopredu určené obdobie
- grafy a prehľady:
  - príjmy alebo výdavky podľa kategórií (koláčový alebo stĺpcový graf)
  - príjmy alebo výdavky podľa mesiacov (dátum (x) / suma (y)
  - prehľady podkategórií (koláčový alebo stĺpcový graf)
  - vytvorenie vlastného grafu podľa zadaných kritérií (dátum od/do, príjem, výdavok, rozdiel, typ grafu)
- ďalšie možnosti:
  - kalkulačka
  - prepočet meny
  - výpočet sprepitného
  - Výpočet zľavy a dane
  - Výpočet splatenia dlhov na kreditnej karte

2.2.4 Porovnanie s mojimi cieľami

Mojim hlavným cieľom je vytvoriť prehľadnú a užívateľsky jednoducho ovládajúcu sa aplikáciu. Všetky spomenuté podobné riešenia sa to tiež snažili splniť. Nie každému sa to však podarilo. Väčšina z aplikácií obsahovala okrem základnej funkcionality aj možnosti, ktoré bežný používateľ asi nikdy nevyužije. To by nevadilo, pokiaľ by tým aplikácia nestrácala na prehľadnosti. Bežný používateľ nemá čas a chuť zisťovať a učiť sa nanovo ovládať aplikáciu. Je zvyknutý na určité ovládacie prvky a ich prehľadné rozloženie. Napríklad pri aplikácii Budgetpulse je problém vykonať aj tú najzákladnejšiu akciu akou je pridanie novej transakcie. A keď to užívateľ konečne nájde, zistí, že rovno pod sebou ma 2 odkazy na pridanie transakcie pričom oba vykonajú rovnakú akciu. Toto riešenie je ideálne ak sa tvorcovia snažia ešte viac domýliť používateľa. A toto je len jedna z množsta vecí, ktoré sa dajú urobiť oveľa lepšie. V jednoduchosti je krása.

Druhou veľkou zmenou oproti ostatným riešeniam bude vytvorenie mobilnej webovej verzie mojej aplikácie. Existujúce riešenia boli zamerané vždy na určitý segment. Išlo vždy buď o webovú, desktopovú alebo mobilnú aplikáciu. Mobilná aplikácia nebola prepojená s webovým riešením a opačne. Tým, že vytvorím aj mobilnú webovú verziu odvodenú od klasickej webovej aplikácie bude tento problém vyriešený. Poslednou veľkou zmenou bude možnosť importu bankových výpisov, čím používateľovi uľahčím zadávanie nových transakcií do aplikácie. Túto možnosť u nás neposkytovala žiadna zo skúšaných aplikácií.

2.3 Model–View–Controller

Pri tvorbe aplikácie budem využívať pravidlá softvérovej architektúry Model–view–controller (MVC) [16] . Ide o spôsob architektúry pri ktorom je oddelený dátový model aplikácie(model), používateľské rozhranie aplikácie(view) a riadiaca logika(controller) do 3 komponentov. Táto architektúra zabezpečuje, že úprava niektorého komponentu má len minimálny vplyv na ostatné komponenty. Tým dosahujeme možnosť nezávislého vývoja, testovania a upravovania každého komponentu samostatne.

Hoci MVC architektúra neeliminuje všetky problémy, poskytuje nám základný pevný bod od ktorého sa môžeme odraziť.

2.3.1 Model

Model je dátovým a funkčným základom celej aplikácie. Pozostáva z aplikačných dát a biznis logiky. Komunikuje s databázou a spracováva dáta. Model vôbec nevie o výstupe a tak nemá s konkrétnou prezentáciou nič spoločné. Funguje tak, že prijme parametre z vonku, spracuje ich a potom pošle dáta von.

Framework CakePHP používa ORM(Objektovo relačné mapovanie) pri ktorom korešpondujú modely s databázovými tabuľkami. Máme teda napríklad model `Clanok` alebo `Komentar`. Inštancie modelov obsahujú atribúty z databázy. Článok má napríklad atribút názov. Triede taktiež môžeme definovať metódy inštancií.

Asi najväčšia výhoda v použití modelu je jednoduchá úprava tabuliek v databáze. Ak by sme ručne písali SQL dopyty, tak by sme v každom z nich museli upraviť zoznam stĺpcov tak, ako sme ich upravili v databáze. Model nám v tomto zásadne pomôže. Postačí nám spraviť úpravu len v ňom.

2.3.2 View

View zabezpečuje zobrazenie výstupu pre používateľa. View získava svoje dáta na zobrazenie priamo z modelu. Model však na komponente View nie je závislý. Používa sa tu však návrhový vzor Observer(Pozorovateľ), ktorý napomáha komponentu Model informovať ostatné komponenty o zmenách dát. V takomto prípade sa komponent View prihlási komponentu Model ako príjemca týchto informácií. V jednoduchosti sa to dá zhrnúť tak, že Observer sa používa pre View, keď je potrebné sledovať Model.

2.3.3 Controller

Controller(radič) je časť MVC architektúry ktorá reaguje na udalosti, ktoré väčšinou pochádzajú od užívateľa a postará sa o zmeny v modely a View. Stará sa o tok udalostí v aplikácii a aplikačnú logiku. Controller teda prijíma všetky zmeny od užívateľa a postará sa o ich vykonanie. Ak vykonaná akcia požaduje zmenu údajov, požiada o to model. Ten podľa vstupných parametrov prevedie požadovanú akciu a upozorní na to View. View zobrazí upravené dáta pre užívateľa.

Pri niektorých typoch implementácie MVC môže Controller priamo komunikovať s View. Controller môže v tomto prípade rozhodovať ktoré View sa majú zobraziť. Jeho hlavnou schopnosťou však naďalej ostáva schopnosť Controlleru upraviť model.

Pri webovej implementácii MVC frameworku sa Controller najčastejšie skladá z dvoch hlavných častí. Prvou časťou je front controller. Tento Controller zachytí všetky http požiadavky, spracuje ich a potom pošle konkrétnym Controllerom. Druhá hlavná časť sú tieto konkrétne Controllery. Controller prijme dáta, ktoré pôvodne pochádzali z http pôžiadavky, uloží ich do Modelu a ten nasmeruje na špecifický View. Ten tieto spracované dáta zobrazí pre používateľa.

2.3.4 MVC verzus Passive View MVC

Ako som už spomínal vo východiskách, moja práca bude založená na frameworku CakePHP. V predchádzajúcej časti som opisoval časti z ktorých je zložená MVC architektúra. CakePHP a podobné webové frameworky však nepoužívajú klasickú MVC architektúru. Sú založené na od nej odvodenej architektúre s názvom PMVC (Passive Model View Controller). Základný rozdiel je v tom, že View v tomto modeli len pasívne zobrazuje dáta. View môže byť šablóna napísaná v niektorom zo šablónovacích jazykov. Samotné PHP, ktoré používa CakePHP je tiež šablónovací jazyk. Často sa však používajú aj iné:

Smarty [15] je jeden z najznámejších šablónovacích jazykov. Je to samostatná šablónovacia knižnica. Smarty nenahrádza PHP, ale len ho dopĺňa. Jeden z jeho hlavných kladov je jednoduchá syntax.

Savant [2] je ďalším známym šablónovacím jazykom. Ako šablónovací jazyk je v ňom použité PHP. Tento jazyk nekompiluje, pretože skripty šablón sú napísané v PHP.

Latte [14] je u nás veľmi známy šablónovací jazyk, ktorý používa český framework Nette. Dovoľuje nám vytvárať a používať vlastné makrá. Je rýchlejší ako Smarty a obsahuje viac možností. Taktiež kladie vysoký dôraz na bezpečnosť. Latte automaticky *escapuje* vypisované premenné.

Spoločnými znakmi týchto šablónovacích jazykov je ich jednoduchá syntax a preklad do jazyka PHP. Ak by sa šablóny prekladali vždy až pri zobrazení aplikácie, boli by zbytočne príliš náročné na výpočet.

View je najčastejšie phtml šablóna, ktorá obsahuje stránku a tagy určitého značkovacieho jazyka. Vďaka tomu sa už nemusí aktualizovať podľa stavu modelu a ani na

neho už nie je priamo napojený. Taktiež nám to dovoľuje zamerať testovanie na Controller bez toho aby sme mali problémy s View. Z tohto vyplýva, že jedným z hlavných kladov tejto architektúry je zlepšenie a zjednodušenie testovateľnosti. [9] Kvôli tomuto modelu však vzniká Controlleru ďalšia povinnosť. Musí synchronizovať View a Model.

Implementácia PMVC v CakePHP prebieha nasledovne:

1. na HTTP server je používateľom zaslaná požiadavka cez GET alebo POST dáta a s nimi spojenú URL adresu
2. server túto požiadavku prepošle ďalej dispečerovi, ktorý je implementovaný na vzore Front Controlleru
3. Front Controller ju následne spracuje a zaistí akciu na konkrétnom Controlleri
4. Controller tieto dáta spracuje podľa aplikačnej logiky a vykoná zmeny na Modeli
5. tieto zmeny sa uložia v databáze
6. Controller si vypýta od Modelu aktualizované dáta a zašle ich na View
7. používateľovi sa zobrazí odpoveď.

2.4 Návrhové vzory

V živote sa často stretávame s úlohami a problémami, ktoré musíme vyriešiť. Väčšinou nie sme prví, kto natrafil na konkrétny problém. Našlo sa už veľa ľudí pred nami, ktorí ho vyriešili. A nie je predsa potrebné vymýšľať koleso znovu a znovu. Platí to aj pri tvorbe aplikácií. Často natrafíme na rovnaké, prípadne aspoň podobné problémy. Takéto situácie nám pri programovaní pomáhajú riešiť návrhové vzory [5] . Návrhový vzor je overená šablóna alebo popis riešenia problému.

Zvyčajne nám objektovo orientované návrhové vzory definujú vzťahy medzi objektmi a triedami. Nedefinujú nám implementáciu konkrétnej triedy.

Pozitív použitia návrhových vzorov je pre nás mnoho. Umožňujú nám napísať kód v ktorom sa aj iný programátor dokáže ľahko orientovať. Sú to už časom zabehnuté funkčné spôsoby a tak je menšia šanca, že spravíme chybu.

Pre objektívnosť spomeniem aj pár negatív. Niekedy sa pozitívne vlastnosti veľmi ľahko zmenia na negatívne. Podľa definície vyžaduje použitie návrhových vzorov dodržiavanie určitých pravidiel pri písaní kódu. Často by sme na implementáciu niektorých funkcií mohli použiť skratku, ale keďže používame návrhové vzory budeme to musieť urobiť tou dlhšou cestou. Občas sa preto oplatí použiť vlastné riešenie.

Kapitola 3

Analýza a návrh

3.1 Funkčné požiadavky

3.1.1 Webová verzia aplikácie

Funkčnosť webovej aplikácie sa líši z pohľadu rôznych typov používateľov:

- Administrátor
- Prihlásený používateľ
- Neprihlásený používateľ

Administrátor

Administrátor môže:

- Mazať používateľov
- Resetovať heslá po žiadosti používateľa

Prihlásený používateľ

Prihlásený používateľ má možnosť kompletnej správy svojich príjmov a výdavkov. Prihlásený používateľ môže:

- Pridávať, editovať a mazať jednorazové príjmy a výdavky
- Pridávať, editovať a mazať opakované príjmy a výdavky

- Možnosť zmeniť jednorazový príjem alebo výdavok na opakovaný a opačne
- Triediť príjmy a výdavky podľa rôznych kategórií, času, podľa toho či sú opakované alebo jednorazové
- prezerať prehľad vývoja stavu financií v čase pomocou grafov
- prezerať predpokladaný vývoj príjmov a výdavkov na nasledujúci mesiac/rok na základe priemerovania za predchádzajúce obdobie
- prezerať odhady či je vzhľadom na predpokladané príjmy a výdavky možné v budúcnosti splatiť plánované a predpokladané výdavky
- vidieť informácie o tom koľko potrebuje ušetriť na výdavkoch, prípadne navýšiť príjmy pre úspešné zvládnutie platenia budúcich výdavkov
- importovať bankou zaslané výpisy transakcií, ktoré následne aplikácia spracuje a priradí základné kategórie k známym transakciám
- editovať a mazať príjmy a výdavky importované z výpisov transakcií banky
- editovať svoje heslo a e-mail v nastaveniach svojho účtu

Neprihlásený používateľ

Neprihlásený používateľ môže:

- registrovať sa alebo sa prihlásiť pokiaľ je už registrovaný
- požiadať admina o zresetovanie svojho hesla v prípade jeho zabudnutia

3.1.2 Mobilná webová verzia aplikácie

Funkčnosť mobilnej verzie aplikácie sa líši z pohľadu 2 typov používateľov.

- Prihlásený používateľ
- Neprihlásený používateľ

Prihlásený používateľ

Prihlásený používateľ môže:

- Pridávať, editovať a mazať jednorazové príjmy a výdavky
- Pridávať, editovať a mazať opakované príjmy a výdavky
- Prezerať prehľad vývoja stavu financií
- Triediť príjmy a výdavky podľa rôznych kategórií, času, podľa toho či sú opakované alebo jednorazové

Neprihlásený používateľ

Neprihlásený používateľ môže:

- registrovať sa alebo sa prihlásiť pokiaľ je už registrovaný
- požiadať admina o zresetovanie svojho hesla v prípade jeho zabudnutia

3.2 Používateľské rozhranie

Ako som spomenul už v časti Porovnanie s mojimi cieľami, v aplikácii Domáce účtovníctvo sa snažím implementovať príjemné, intuitívne a hlavne jednoduché používateľské rozhranie. Na vrch stránky som umiestnil navigáciu so všetkými potrebnými základnými položkami. Pod navigáciou sa v ľavej časti stránky nachadza dodatočné menu s odkazmi na konkrétnejšie podstránky. Okrem nich sa tam nachádzajú aj akcie ktoré možno aplikovať na aktuálnej stránke. Napríklad pri prehľade konkrétnej transakcie sú tam zobrazené možnosti na jej editáciu alebo prípadne na jej zmazanie. Ako som spomenul táto ponuka sa mení v závislosti od vybranej stránky. Pri prehľadoch transakcií sa v tejto časti nachádza nastavenie filtra, podľa ktorého sa zobrazia transakcie.

Napravo od tohoto menu sa nachádza obsah stránky. Pri prehľadoch ide zvyčajne o graf pod ktorým je umiestnená tabuľka s možnosťou triedenia podľa stípcov a jednoduchého stránkovania.

3.3 Databázový model

3.3.1 Transactions

Pre aplikáciu Domáce účtovníctvo sú najdôležitejšou časťou transakcie. Rovnako je pre databázový model najvýznamnejšia tabuľka `transactions`. Táto tabuľka uchováva informácie o všetkých transakciách. Transakcie sú rozdelené na príjmy a výdavky podľa `transaction_type_id` a preto na to nie je nutné 2 špeciálne tabuľky. `Transactions` obsahuje cudzie kľúče na stĺpce v tabuľkách `transaction_types`(`transaction_type_id`), `categories`(`category_id`), `subcategories`(`subcategory_id`), `users`(`user_id`) a dokonca aj sama na seba. To využíva pri opakovaných transakciách odkazovaním na svoj stĺpec `original_transaction_id`.

3.3.2 Users

Druhou významnou tabuľkou(modelom??) sú users(používatelia?), ktorí sú reprezentovaní v tabuľke `users`. Nájdeme v nej všetky informácie týkajuce sa používateľa. Zaujímavá je položka active. Určuje či je používateľ stále aktívny a má prístup do systému. V prípade zrušenia konta sa zmení len táto položka a tým je používateľovi zablokovaný ďalší prístup do aplikácie. Táto tabuľka obsahuje jediný cudzí kľuč `user_type_id`, ktorý sa vzťahuje na stĺpec `id` v tabuľke `user_types`. Ďalej je v tejto tabuľke zadefinovaný vzťah hasMany na tabuľky `transactions`, `categories`, `subcategories` a `imports`. Z toho vyplýva, že jednému používateľovi môže prislúchať viacero záznamov z týchto tabuliek.

3.3.3 Categories

Jednoduchá tabuľka obsahujúca informácie o kategóriách transakcií. Obsahuje len jeden cudzí kľúč `user_id` na na stĺpec `id` v tabuľke `users`.

3.3.4 Subcategories

Tabuľka obsahujúca informácie o podkategóriách. Na rozdiel od `Categories` obsahuje okrem cudzieho kľúča aj ďalší cudzí kľúč `category_id` odkazujúci na kategóriu ktorej patrí.

3.3.5 User_types

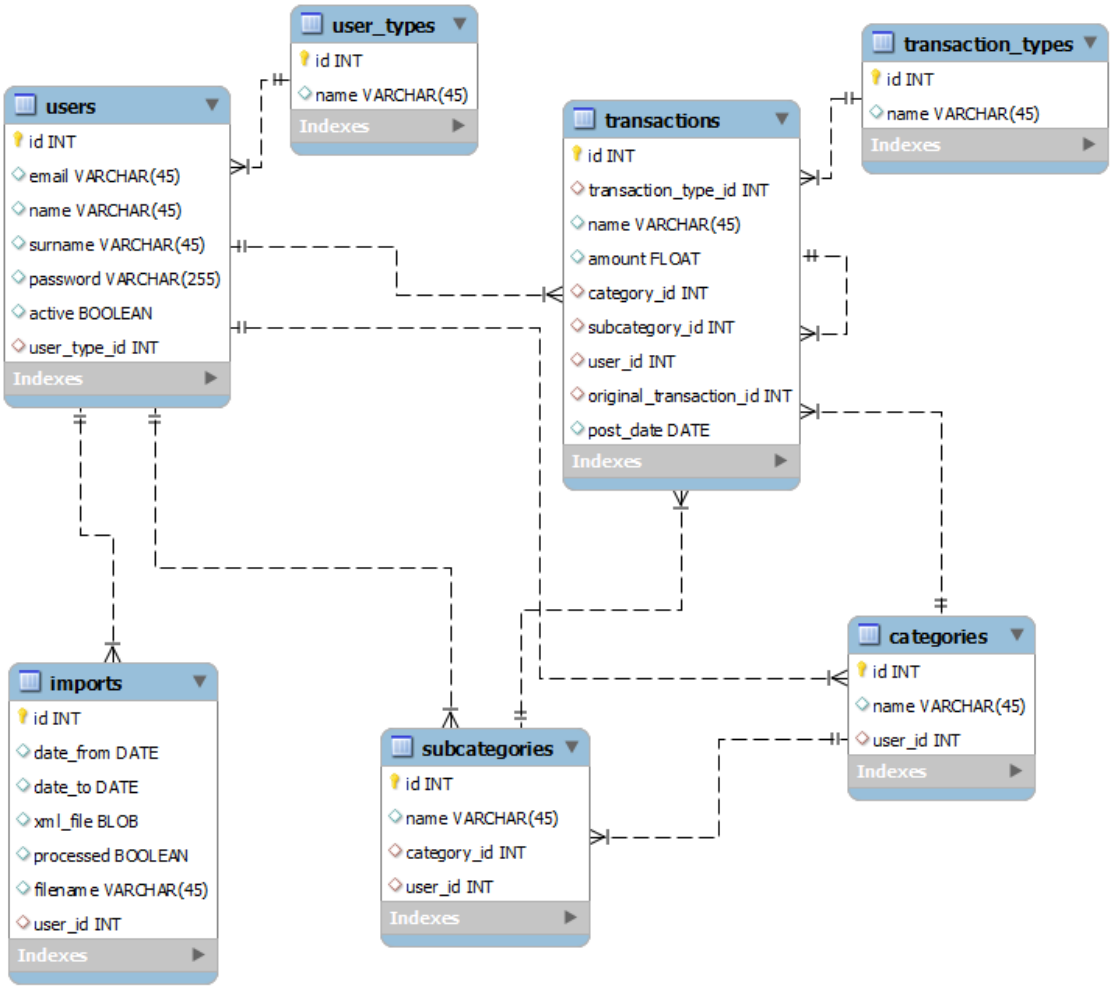
Tabuľka `user` types slúži na identifikáciu a pomenovanie typu používateľa. Aktuálne sú možné 2 typy používateľov. Admin a bežný používateľ.

3.3.6 Transaction_types

Tabuľka `transaction_types` slúži na identifikáciu a pomenovanie typu transakcie. Tie sú rozdelené na príjmy a výdavky.

3.3.7 Imports

Tabuľka obsahujúca len jeden cudzí kľúč `user_id`, ktorý definuje vzťah `belongsTo` k tabuľke `user` a konkrétnemu userovi(používateľovi).



Obr. 3.1: Domáce účtovníctvo - Databázový model

Kapitola 4

Implementácia

TODO

Vystupy pre highcharts plugin - rozne typy prehladov podla casovych obdobi (den, mesiac, rok)

Kapitola 5

Záver

Zaverecny popis.

Literatúra

[1] B. Bernard. Úvod do architektury MVC, 2009.
http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/.

[2] B. Bieber. Savant, 2013.
http://phpsavant.com/.

[3] Bishinews. Expense manager, 2013.
https://play.google.com/store/apps/details?id=com.expensemanager.

[4] Bookmark. Money lover, 2013.
https://play.google.com/store/apps/details?id=com.bookmark.money.

[5] R. Bouda. Seriál návrhových vzorů, 2012.
http://programujte.com/clanek/2012032900-serial-navrhovych-vzoru-1-dil/.

[6] Buxfer. Buxfer, 2013.
https://www.buxfer.com/.

[7] CakePHP. CakePHP documentation, 2013.
http://cakephp.org/pages/documentation.

[8] D. Driven. Highcharts plugin, 2013.
https://github.com/destinydriven/cakephp-high-charts-plugin.

[9] M. Fowler. Passive view, 2006.
http://martinfowler.com/eaaDev/PassiveScreen.html.

[10] Handy Apps Inc. Easy money, 2013.
https://play.google.com/store/apps/details?id=com.handyapps.easymoney.

[11] Highsoft Solutions AS. Highcharts, 2013.
http://www.highcharts.com/.

[12] L. Hunt. A preview of HTML 5, 2007.
http://alistapart.com/article/previewofhtml5.

[13] iSquare Inc. Budget pulse, 2013.
https://www.budgetpulse.com/.

[14] Nette Foundation. Latte šablóny, 2013.
http://doc.nette.org/cs/templating.

[15] New Digital Group, Inc. PHP template engine smarty, 2013.
http://www.smarty.net/.

[16] sdraco. MVC architektura.
http://www.devbook.cz/mvc-architektura-navrhovy-vzor.

[17] S. Svetlík. RQ money, 2013.
http://www.rq.sk/.

[18] W3C. Cascading style sheets level 2 revision 1 (CSS 2.1) specification, 2011.
http://www.w3.org/TR/CSS2/.

[19] W3C. HTML 5.1 nightly, 2013.
http://www.w3.org/html/wg/drafts/html/master/single-page.html.

Príloha

Priložené DVD médium

Priložené DVD obsahuje:

- Text tejto práce vo formáte TEX a PDF
- Zdrojové súbory aplikácie Domáce účtovníctvo:
  - *nieco* - nieco