UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Domáce účtovníctvo

Bakalárska práca

2013 **Peter Kotulič**

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Domáce účtovníctvo

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
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 31. mája .....

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 [20] 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 elementmi 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.

Ďalšou 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 [19] (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]

Webová aplikácia Budget Pulse ponúka jednoduchú administráciu a prehľady transakcií. Trochu horšie je spracované menej prehľadné pridávanie výdavkov a príjmov. Jednou za základných funkcií tejto aplikácie je vytvorenie novej 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 transakcie a zadá prípadné opakovania transakcie. Zaujímavou funkciou v tejto aplikácii je možnosť rozdelenia transakcie na menšie časti a ich priradenie do rôznych kategórií. Budget Pulse taktiež ponúka vytvorenie viacerých kont pod jedným účtom. Ďalšou užitočnou funkciou je import a export dát z aplikácie. Za spomenutie taktiež stojí 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. Na jednoduchý prehľad medzi transakciami a kategóriami využíva aplikácia okrem klasického tabuľkového zobrazenia kategórií v čase aj grafy. Jednoduchý koláčový graf kategórií a stĺpcový graf kategórií za určité obdobie.

Buxfer [6]

Druhá podobná webová aplikácia ktorú som testoval bola Buxfer. Hneď na prvý pohľad ma zaujal oveľa prepracovanejší a prehľadnejší dizajn ako pri Budget Pulse. Hlavne veľmi dobre spracovaná hlavná stránka. Zobrazuje 4 panely. Aktuálny stav účtu, výdavky za posledný mesiac zobrazené podľa kategórií koláčovým grafom a taktiež stĺpcovým grafom. Tieto grafy sa dajú v menu reports filtrovať aj podľa iného časového rozmedzia. Na poslednom panely sa nachádzajú pripomienky o budúcich výdavkoch. Pri transakciách nájdeme ich tabuľkové zobrazenie s popisom, sumou, tagmi, účtom a dátumom transakcie. Zaujímavou funkcie je prehľad požičaných peňazí. Pri pridávaní transakcií označujeme či ide o príjem, výdavok, transfer alebo refund. Do aplikácie môžeme taktiež nahrávať výpisy z banky. Za zmienku stojí aj vypočítavanie prehľadov do budúcna na základe predchádzajúcich príjmov a výdavkov.

2.2.2 Desktopové aplikácie

RQ Money [17]

Jedinou desktopovou aplikáciou ktorá sa mi páčila a stála za otestovanie bolo RQMoney od slovenského autora. Aplikácia je vytvorená v Delphi. V porovnaní s predchádzajúcimi aplikáciami je RQMoney oveľa komplexnejšia a ponúka oveľa väčšiu funkcionalitu. V tomto sa dosť odlišuje od mojej webovej aplikácie Domáce účtovníctvo. Tá ma za cieľ prehľadnosť a jednoduchosť.

RQMoney ponúka klasickú evidenciu príjmov, výdavkov a presunov. Používateľovi sú dostupné rôzne typy filtrov a radení podľa vlastného výberu. Užitočná funkcia je plánovanie transakcií s možnosťou prezerania platobného kalendára. S tým súvisí aj možnosť porovnania plánovaných a skutočných transakcií za mesiac. Štatistiky ponúkajú podrobné prehľady za rôzne časové obdobia vďaka tabuľkám a grafom. Používateľ môže v aplikácii použiť vlastné SQL príkazy. RQMoney obsahuje aj prehľadné tlačové zostavy a exporty tabuliek do MS Excel. Rovnako je možné exportovať aj grafy.

2.2.3 Mobilné aplikácie

Zaoberal som sa len aplikáciami pre systém Android

Easy Money - Android [10]

Prvá aplikácia pre Android ktorú som testoval bola Easy Money. Ponúka rôzne typy účtov. Napríklad cash, kreditná karta, účet v banke. Pri pridávaní transakcií určujeme jej názov, cenu, kategóriu, dátum, či ide o príjem alebo výdavok a opakovanie. Okrem toho máme možnosť rozdeliť transakciu do viacerých kategórií a taktiež určiť, že ide len o prevod medzi účtami.

V prehľadoch nám aplikácia ponúka zobrazenie príjmov alebo výdavkov podľa kategórií v stĺpcových grafoch. Okrem nich aj denne porovnanie príjmov a výdavkov a tiež ich mesačný zoznam a sumy za konkrétne mesiace. Ďalšou užitočnou vlastnosťou je pripomienkovač platenia faktúr a účtov. Zaujímavou funkciou je aj nastavenie mesačného rozpočtu a prehľad jeho využitia počas mesiaca. Easy Money podporuje import a export do QIF a CSV. Medzi transakciami môže používateľ prehľadávať pomocou keywords.

Money Lover - Android [4]

Money Lover je najprehľadnejšia a najintuitívnejšia mobilná aplikácia, ktorú som testoval. Obsahuje jednoduché pridávanie transakcií pri ktorom určujeme či ide o príjem, výdaj, pôžičku alebo dlh. Potom vypĺňame polia názov, kategória, suma, poznámka, dátum, účet a prípadne nastavenie opakovania. Pri opakovaniach sa nedá nastaviť konkrétny dátum, ale len začiatok alebo koniec týždňa, pripadne mesiaca. Ako je možné už vidieť z vyplnených polí, môžeme používať rôzne typy účtov. Transakcie môžeme prehliadať v tabuľkových prehľadoch s možnosťou triedenia podľa dátumov, kategórií alebo príjmu a výdavkov. Koláčový graf kategórií za určité obdobie je už samozrejmosť. Aplikácia obsahuje aj menej zvyčajné funkcie ako je správa dlhov, prepočet meny, výpočet vrcholu, vyhľadanie bankomatu, banky a počítanie úroku. Zaujímavou funkciou je aj sprievodca pri nastavení aplikácie, ktorý by sa zišiel hlavne pri ostatných menej prehľadných aplikáciách.

Expense Manager - Android [3]

Poslednou Android aplikáciou ktorú som testoval bol Expense Manager. Klasicke pridávanie príjmov a výdavkov ma táto aplikácia vylepšené o možnosť prednastavenia predvypĺňaných polí. Ponúka taktiež možnosť pridávania opakujúcich sa transakcií. Na dopredu určené opakujúce sa obdobie má používateľ možnosť nastaviť rozpočet.

Prehľad v transakciách ponúkajú koláčové a stĺpcové grafy príjmov alebo výdavkov triedených podľa kategórií a podľa mesiacov. Podkategórie si tiež môžeme zobraziť na koláčovom alebo stĺpcovom grafe. Graf je možné si vytvoriť aj zadaním vlastných kritérií(dátum od/do, príjem, výdavok, rozdiel, typ grafu). Expense Manager ponúka ešte aj ďalšie možnosti ako je kalkulačka, prepočet meny, výpočet sprepitného, výpočet zľavy a dane alebo výpočet splatenia dlhov na kreditnej karte.

2.2.4 Porovnanie s mojimi cieľmi

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 množstva 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 požiadavky, 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

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 meno a priezvisko v nastaveniach svojho účtu
- požiadať o resetovanie hesla do aplikácie v nastaveniach svojho účtu

Neprihlásený používateľ

Neprihlásený používateľ môže:

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

3.2 Používateľské rozhranie

3.2.1 Webové rozhranie

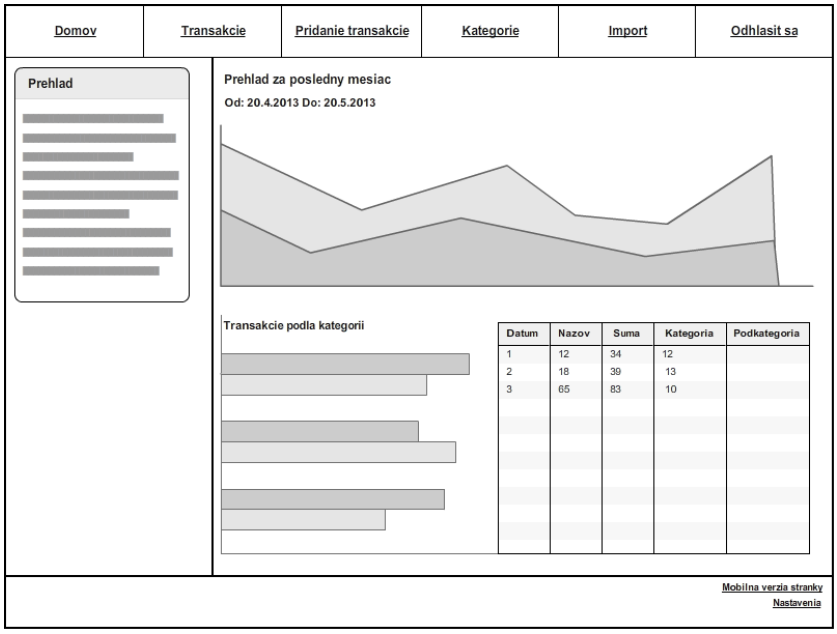
Ako som spomenul už v časti Porovnanie s mojimi cieľmi, v aplikácii Domáce účtovníctvo sa snažím implementovať príjemné, intuitívne a hlavne jednoduché používateľské rozhranie. Na vrchnú časť stránky som umiestnil navigáciu so všetkými potrebnými základnými položkami. Vľavo pod navigáciou sa nachádza aktuálna bilancia financií používateľa. Prvý je aktuálny stav zobrazujúci aktuálnu sumu jeho financií. Pod ním sa nachádza suma za plánované výdavky na najbližšie 3 mesiace. Nakoniec sú to sumy príjmov a taktiež aj výdavkov za posledný mesiac. Pokiaľ to je potrebné, tak sa pod týmto prehľadom zobrazuje dodatočné menu s odkazmi na konkrétnejšie podstránky. Okrem nich sa na tomto mieste 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 aktuálne vybranej stránky. Pri prehľadoch transakcií sa v tejto časti zobrazí aj nastavenie filtra, podľa ktorého sa vyberú transakcie. Na výber je rozsah

dátumov od a do a taktiež môže byť dostupná možnosť rozdelenia transakcií v grafe podľa rokov, mesiacov alebo dní.

Napravo od tohto menu sa nachádza obsah stránky. Pri prehľadoch ide zvyčajne o graf pod ktorým je umiestnená tabuľka s možnosťou triedenia podľa rôznych stĺpcov a jednoduchého stránkovania. Nižšie rozoberiem stránky so základnej ponuky navigácie.

Domov

Základná domovská stránka je trochu špecifickejšia. Používateľovi sa snažím poskytnúť všetky základne informácie bez potreby skrolovania po stránke. Nachádza sa tu základný graf príjmov a výdavkov rozdelených podľa dní za poslední mesiac. Na x-ovej súradnici sa nachádzajú dni a na y-ovej suma. Výdavky nadobúdajú mínusové hodnoty. Príjmy smerujú do plusových hodnôt. Sumy za konkrétny deň sa vyrátané sčítaním výdavkov, respektíve vkladov. Okrem týchto údajov sa v grafe nachádza aj krivka s priebežnou bilanciou. Tá je počítaná od vybraného časového obdobia a pri každom vklade, prípadne výdavku, zobrazuje aktuálnu bilanciu financií za toto obdobie. Z toho vyplýva, že krivka pri veľkých výdavkoch a nedostatku financií môže smerovať aj do mínusových hodnôt. Pod každým grafom sú umiestnené jeho ovládacie prvky na ktorých si používateľ vyberie ktoré údaje chce zobraziť. Pri tomto sú to príjmy za dni, výdavky za dni a pribežná bilancia.

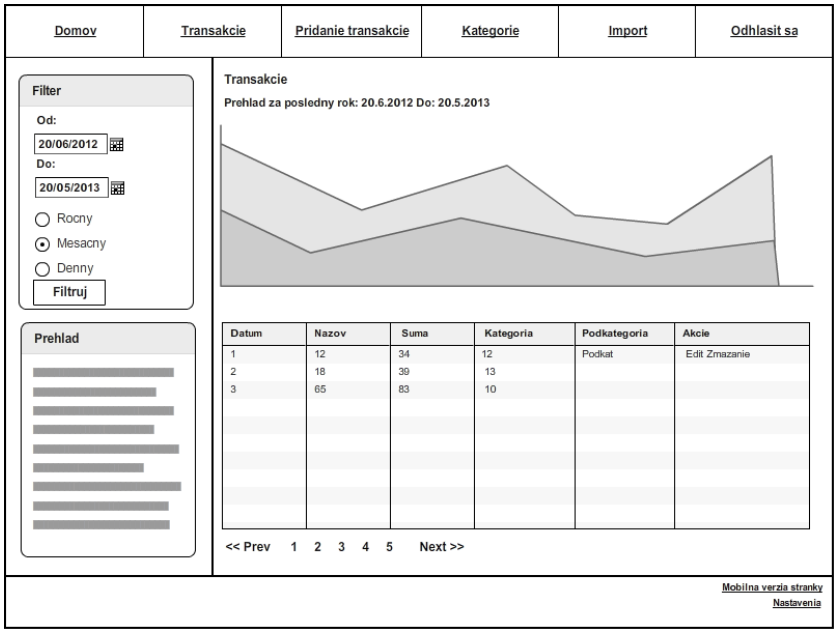


Obr. 3.1: wireframe - domov

Hneď pod ním sa nachádza ďalší graf zobrazujúci celkové sumy v rôznych kategóriách za rovnaké mesačné obdobie. Tento graf je má však len polovičnú šírku vrchného grafu a tak sa vedľa neho zmestil aj tabuľkový rozpis posledných transakcií aj s možnosťou stránkovania a usporiadania podľa rôznych stĺpcov. Ako som už v úvode naznačil, všetky tieto informácie uvidí používateľ bez nutnosti skrolovania. Pôvodne som chcel tieto informácie s kategóriami zobrazovať v koláčovom grafe. Po dohode s pánom školiteľom som od toho upustil, pretože použitý graf je prehľadnejší a v koláčovom grafe ma človek väčší problém porovnať uhly a zistiť ktorý je väčší. Pri aktuálnom grafe to je možné rozpoznať ihneď.

Transakcie

Hlavným zámerom tejto stránky je používateľovi sprostredkovať prehľad vo všetkých transakciách a grafické porovnanie príjmov a výdavkov v určitom období. Pod navigáciu som umiestnil graf podobný tomu z domovskej stránky. Obsahuje však niekoľko rozdielov. Príjmy aj výdavky sú v ňom zobrazované do plusových hodnôt. Tým je zjednodušené ich porovnávanie. Narozdiel od krivky s priebežnou bilanciou sa tu nachádza rozdielová krivka. Pri každom dni, prípadne mesiaci alebo roku, zobrazuje rozdiel príjmov výdavkov. Ak teda v určitý deň(mesiac, rok) presiahne suma výdavkov sumu príjmov, tak bude táto krivka v mínusových hodnotách.



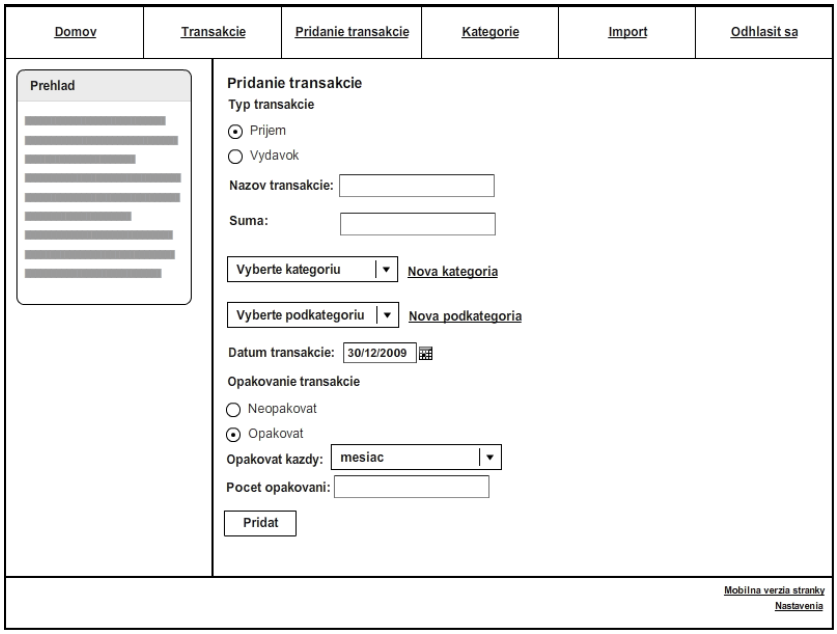
Obr. 3.2: wireframe - transakcie

Pod grafom sa nachádza tabuľka s rozpisom transakcií. Okrem údajov ktoré ponúka podobná tabuľka z domovskej stránky obsahuje táto naviac aj tlačidlá s akciami, ktoré sa dajú použiť pri každej transakcii. Ide o editovanie transakcie, vymazanie transakcie a posledným je vymazanie vybratej transakcie a zároveň všetkých jej ďalších opakovaní v budúcnosti. Jej predchádzajúce opakovania zostanú uložené.

Všetky údaje pre grafy a transakcie je možné filtrovať pomocou ponuky v ľavej časti stránky. Používateľ si vyberie časový rozsah od-do a rozdelenie v grafe podľa rokov, mesiacov alebo dní.

Pridanie transakcie

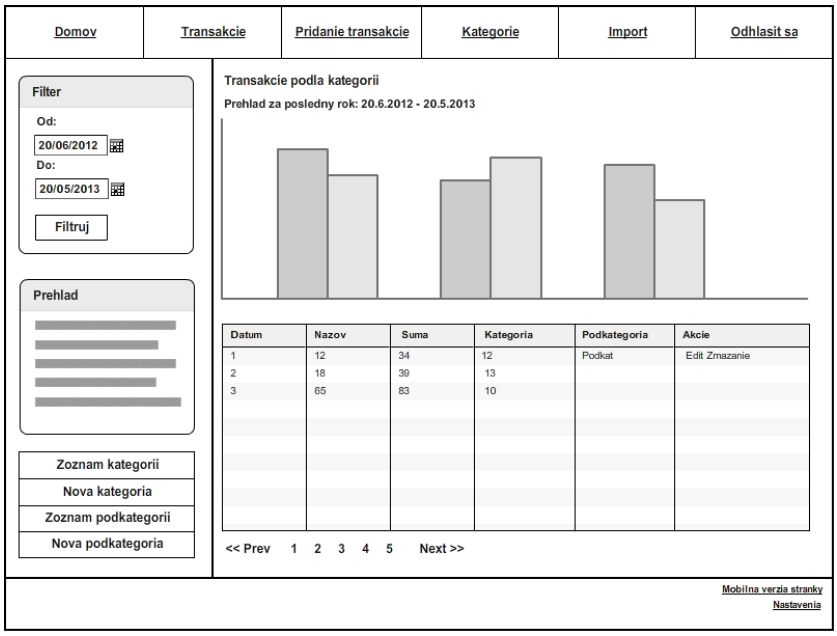
Jednoduchá stránka na pridávanie transakcií. Používateľ vyberá typ transakcie(príjem alebo výdavok). Názov transakcie, sumu, kategóriu, podkategóriu ktorá je zreťazená s kategóriou a dátum transakcie. Nakoniec má možnosť vybrať či pôjde o jednorazovú alebo opakovanú transakciu. Pri vybratí opakovanej sa mu zobrazia ďalšie možnosti. Interval opakovania a počet opakovaní. V tomto prípade sa vytvorí jedna pôvodná transakcia a ďalšie jej opakovania, ktoré budú v obsahovať pole `original_transaction_id` s `id` ich pôvodnej transakcie.



Obr. 3.3: wireframe - pridanie transakcie

Kategórie

Základný graf na tejto stránke zobrazuje príjmy a výdavky podľa kategórií za vybraté obdobie pomocou filtra. Prednastavené obdobie je rok dozadu od aktuálneho dátumu. Pod ním sa nachádza tabuľkový výpis transakcií ako na stránke *Transakcie*. Na tejto stránke je taktiež možné použiť filter na rozsah dátumov od-do. Pri vybratí konkrétnej kategórie je používateľovi zobrazený graf kategórie v čase. Tento graf je tiež možné filtrovať. V spodnej časti stránky sú vtedy zobrazené aj podkategórie ktoré patria do vybranej kategórie.



Obr. 3.4: wireframe - kategorie

Po vybratí podkategórie sa zobrazí používateľovi informácia o nadradenej kategórii a taktiež tabuľka so všetkými transakciami, ktoré patria do tejto podkategórie. Samozrejmosťou sú tlačidlá s akciami, ktoré je možné aplikovať na vybratú podkategóriu. Ide o mazanie a editovanie a nachádzajú sa v ľavej časti stránky.

Import

Stránka import poskytuje používateľovi tabuľkový prehľad importovaných výpisov z banky. Pri každom výpise sú na výber akcie spracovať, editovať a vymazať výpis. Najdôležitejšia akcia je spracovanie výpisu. Spracovanie výpisu podporuje aktuálne len výpisy Slovenskej Sporiteľne vo formáte ABO. Importovaný výpis z banky je spracovaný

a používateľ dostane stránku s dodatočnou možnosťou upraviť kategórie a podkategórie každej transakcie z importovaného výpisu.

Nastavenia

Základné nastavenia účtu používateľa. Na tejto stránke si má používateľ možnosť zmeniť meno, priezvisko alebo môže požiadať o zmenu svojho hesla. Z dôvodu zvýšenej bezpečnosti mu bude potom zaslaný email s resetovacím odkazom. Vzhľadom k tomu, že táto ponuka nastavení nebude veľmi často využívaná, umiestnil som odkaz s ňou na spodnú časť stránky do pätičky.

Odhlásiť sa

Používateľ je po kliknutí odhlásený z aplikácie, zároveň je zrušená jeho session a je presmerovaný na prihlasovaciu stránku.

3.2.2 Mobilné rozhranie

Pre zjednodušenie ovládania aplikácie na mobilných zariadeniach som pracoval aj na mobilnom rozhraní. Veľmi nápomocnou mi v tomto bola knižnica jQuery Mobile [18] ktorá mi pomohla okrem funkčnosti aj s grafickým prevedením aplikácie vďaka jej základnému css layoutu pre mobilné prehliadače. Navigáciu som pri tomto rozhraní presunu spodnú časť stránky, tak ako to býva zvykom pri mobilných zariadeniach. Pokiaľ nie je potrebná tak sa automaticky zroluje, aby bola využitá maximálna plocha a mohlo byť zobrazených čo najviac informácií. V tejto spodnej časti stránky sa nachádza aj tlačidlo na prepnutie na plnú verziu stránky a taktiež aj tlačidlo na odhlásenie sa z aplikácie.

Pre zachovanie prehľadnosti a jednoduchosti ovládania som počet odkazov v navigácii znížil na základné 4, ktoré ďalej opíšem trochu podrobnejšie.

Domov

Základná stránka aplikácie, ktorú uvidí používateľ po prihlásení. Zobrazuje prehľadnú aktuálnu bilanciu financií používateľa. Je to ten istý prehľad ktorý nájde používateľ v plnom webovom rozhraní vľavo pod navigáciou. Pod touto bilanciou sa nachádza graf transakcií podľa kategórií. V tomto grafe sú zarátané transakcie za posledný mesiac. Pod ním je prehľadná tabuľka s transakciami. Zobrazuje dátum, názov, sumu a

kategóriu do ktorej je zaradená konkrétna transakcia. Graf zobrazujúci transakcie za určité obdobie som vynechal. Vzhľadom na malé rozlíšenie by nemal dostatočne dobrý informatívny charakter.

Transakcie

Prehľadný tabuľkový zoznam transakcií za posledný mesiac. Možnosť triedenia podľa rôznych stĺpcov je samozrejmosť. Každá transakcia zobrazuje svoj dátum, názov, sumu, kategóriu a podkategóriu. Na rozdiel od zoznamu na domovskej stránke je pri transakcii aj možnosť jej úpravy, zmazania. Taktiež sa tu nachádza aj možnosť zmazania vybranej transakcie a zároveň aj všetkých jej budúcich opakovaní.

Pod touto tabuľkou je umiestnené filtrovacie menu, ktoré som opisoval už pri webovom rozhraní.

Pridanie transakcie

Pridávanie transakcie je nastaveniami zhodné s pridávaním transakcie v plnom webovom rozhraní. Upravený je len dizajn tak aby vyhovoval mobilným zariadeniam.

Kategórie

Stránka kategórie zobrazuje graf príjmov a výdavkov podľa kategórií za posledný rok. Pod grafom za nachádza rozpis týchto kategórií s podrobnými informáciami o každej z nich tak ako je to aj na stránke *Transakcie*.

3.3 Objektový model

Transactions

Pre aplikáciu Domáce účtovníctvo sú najdôležitejšou časťou transakcie. Rovnako je pre objektový model najvýznamnejšia trieda `transactions`. Táto trieda uchováva informácie o všetkých transakciách. Transakcie sú rozdelené na príjmy a výdavky podľa `transaction_type_id` a preto na ne nie sú nutné 2 špeciálne triedy. Každej transakcii je okrem jej typu priradená aj jej kategória, podkategória, id používateľa ktorému patrí a informácia či ide o transakciu ktorá sa opakuje. Takisto má každá transakcia svoj názov, sumu a dátum kedy prebehla.

Users

Druhou významnou triedou sú používatelia, ktorí sú reprezentovaní v triede `users`. Nájdeme v nej všetky informácie týkajúce 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. Každý zaregistrovaný používateľ je zatriedený do jedného z dvoch typov. Môže ísť o admina alebo o bežného používateľa. Každý používateľ môže mať viacero svojich transakcií, kategórií, podkategórií a výpisov.

Categories

Trieda obsahujúca informácie o kategóriách transakcií. Každá kategória patrí určitému svojmu používateľovi. Kategória má minimálne jednu svoju podkategóriu.

Subcategories

Trieda obsahujúca informácie o podkategóriách. Každá podkategória patrí svojmu používateľovi a taktiež aj svojej nadradenej kategórii.

User_types

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

Transaction_types

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

Imports

Trieda, ktorá obsahuje informácie o importoch používateľa. Každý import patrí svojmu konkrétnemu používateľovi.

Kapitola 4

Implementácia

TODO MAYBE info napr o domovskej stranke ...v akom controlleri je atd, ako cake implementuje controller .. ako som includoval controller do ineho

4.1 CakePHP

Základná implementácia vo frameworku CakePHP je na princípe toho, že každá trieda z tabuľky má svoj vlastný `controller`, `model` a rôzne prislúchajúce `views`. Tento princíp som využíval aj ja. V každom controllery som si definoval potrebné funkcie, nazývané aj akcie tohto controllera.

4.1.1 AppController

Všetky mnou vytvorené controllery rozširujú možnosti preddefinovanej triedy `AppController` a tá možnosti základnej triedy `Controller`. V triede `AppController` som si definoval všetky potrebné akcie, triedy a komponenty CakePHP, ktoré budem chcieť používať v ostatných mnou vytvorených controlleroch. U mňa to boli triedy `CakeTime`, `CakeEmail`(nahrádza `EmailComponent` zo starších verzií cakePHP), komponenty `Auth`, `Session` a `Cookie`. Zadefinoval som tu taktiež aj `layout` pre webové a mobilné rozhranie.

Hlavná a veľmi dôležitá funkcia v `AppController` je verejná funkcia `beforeFilter`. Dávajú sa do nej akcie ktoré sú volané ako prvé a ešte pred akciou controllera. Sú to napríklad akcie súvisiace s autorizáciou používateľa a jeho oprávnení. Taktiež v nej kontrolujem prístup z mobilného zariadenia a s tým súvisiace neskoršie prepnutie na

mobilný vzhľad aplikácie.

Ďalšou funkciou je `afterFilter`, ktorá je zavolaná až po akcii controllera a renderovaní `view`. Skontroluje či bol v `beforeFilter` zistený prístup z mobilného zariadenia a nastaví príslušný mobilný vzhľad. Pokiaľ je vytvorené, tak nastaví aj špecifické mobilné `view`.

4.1.2 TransactionsController

Základom aplikácie Domáce účtovníctvo je práca s transakciami a preto je práve `TransactionsController` najzaujímavejším a najdôležitejším controllerom. Zaoberá sa spracovaním transakcií a tvorbe prehľadných grafov a tabuľkových prehľadov. Už skôr som spomínal, že na tvorbu grafov používam Highcharts plugin. Preto bolo jednou z prvých vecí pridanie komponentu `HighCharts.HighCharts` do tohto controllera. Používam ho v každej akcii v ktorej generujem grafy.

4.2 Dáta pre HighCharts grafy

Základom každého grafu sú dáta. V mojich grafoch potrebujem naplniť hodnotami polia pre X-ové súradnice a polia pre Y-ové súradnice. Napríklad na domovskej stránke zobrazuje vrchný graf na X-vej súradnici dni za posledný mesiac a na y-ovej sumy transakcií za tieto konkrétne dni. Polí s x-ovými súradnicami môže byť aj viac. Záleží to podľa počtu údajov ktoré chcem zobrazovať. Napríklad na spomenutej domovskej stránke je v jednom grafe stĺpcový prehľad príjmov, stĺpcový prehľad výdavkov a krivka priebežnej bilancie. Pod grafom sa nachádzajú tlačidlá s možnosťou zobrazenia len niektorého z týchto údajov.

Jednou zo zložitejších častí implementácie týchto grafov bolo správne prednastavenie polí s dátami a nulovými hodnotami. Je to spôsobené tým, že nie každý deň musí byť pridaná určitá transakcia. Preto pokiaľ som nechcel, aby mi z grafu vypadli dni bez transakcií, musel som si po vybratí rozsahu dátumov vo filtry nastaviť nulové hodnoty pre každý deň, prípadne mesiac alebo rok. Najzložitejšie sa to implementovalo pri dennom filtry. Najjednoduchši prípad bol ak bol rozsah dní len v strede určitého mesiaca. Pokiaľ rozsah prekračoval mesiac potreboval som dĺžky všetkých mesiacov ktoré rozsah zahŕňal. Ak rozsah prekročil rok, tak sa príprava dát sťažila ešte viac. Pri filtrovaní dní som takto získaval až trojrozmerné polia naplnené nulovými hodnotami. Až potom

som ich mohol naplniť dátami z reálnych transakcií.

4.3 Stránkovanie a grafy transakcií

Pokračujem opisom domovskej stránky. Pod grafom sa nachádza tabuľka s prehľadom všetkých transakcií za tento posledný mesiac. Na tvorbu týchto tabuliek som využil komponent `PaginatorComponent`. Ide o veľmi dobre spracovaný komponent, ktorý ponúka CakePHP na tvorbu dotazov na databázu. Ponúka možnosť stránkovania a triedenia podľa stĺpcov tabuľky.

```
$this->paginate = array(
        'limit' => 20,
        'order' => array(
                        'Transaction.post_date' => 'asc'
        ),
        'conditions' => array(
                        'Transaction.user_id' =>
                           $this->Session->read('User.id'),
                        'Transaction.post_date >=' => $filterdata['from_date'],
                        'Transaction.post_date <=' => $filterdata['to_date'],
        ),
);
```

Ako je možné vidieť v ukážke kódu, najčastejšie sa v ňom nastavujú 3 možnosti. Limit výpisov na jednu stránku, základné zoradenie týchto výpisov a podmienky výberu. V mojom prípade som nastavil limit 20 transakcií na jednu stránku, triedenie podľa dátumu transakcie a výber transakcií patriacich aktuálne prihlásenému používateľovi v rozsahu od a po určitý dátum. Framework si vďaka takémuto nastaveniu vybral všetky potrebné dáta s transakciami. Pôvodne som používal tieto dáta vybraté cez `paginate` aj pre grafy. Zistil som však, že dáta boli vybraté vždy len pre konkrétnu stránku zo stránkovania a tak sa mi na grafe vygenerovala napríklad len polovica údajov. Keď som chcel ďalšie, bolo potrebné prejsť na druhú stránku v stránkovaní. Vyriešil som to tak, že dáta pre grafy som získaval pomocou funkcie find. Tak som sa dostal ku všetkým potrebným dátam bez toho aby boli obmedzené stránkovaním.

Ďalší problém, ktorý sa vyskytol, bol pri použití filtra od/do a prepínania stránkovania. Ak používateľ použije filter pre výber rozsahu dátumov transakcií a rozdelenia týchto transakcií, zmení týmto údaje zobrazované v grafe a taktiež aj v tabuľke. Ak však potom klikne na inú stránku v stránkovaní, tieto údaje sa stratili a boli nahradené tými pôvodnými. Riešením bolo použitie Session. Po nastavení filtra som údaje z neho uložil do Session. Vďaka tomuto riešeniu sa pri stránkovaní nestratili a mohol som ich používateľovi prednačítať aj v iných častiach aplikácie, kde bola možnosť filtrovať dáta pre zobrazenie v grafoch a tabuľke.

4.4 Spracovanie výpisov

Jednou z ďalších zaujímavých a zložitých prvkov aplikácie bolo nahrávanie a spracovanie výpisov. Aktuálna verzia aplikácie podporuje spracovanie výpisov Slovenskej sporiteľne, ktorá ich poskytuje v zaužívanom formáte ABO. Na pochopenie tohoto ABO výpisu som si najskôr musel naštudovať štruktúru jeho výpisu.

Celý súbor výpisu sa skladá z 2 typov viet. Prvá veta je typu 074. Je to hlavička výpisu z účtu ktorá má fixnú dĺžku 114 znakov. Druhá veta je typu 075. Označuje údajovú položku výpisu z účtu. Jej fixná dĺžka je 128 znakov. Podrobnejšia štruktúra týchto viet zaberá 2 strany a preto ju tu celú nebudem opisovať. Zaujímavosťou v tomto výpise je, že jednotlivé pozície čísla účtu v tomto výpise sú pozmenené. Napríklad pre číslo účtu 0123456789 je číslo účtu v ABO výpise v tvare 9785012346. Takto vyzerá ukážka spracovania tohto čísla účtu:

```
$cislo_uctu_partnera = substr($lines[$i], 3, 10); // nacitanie cisla uctu
    zo spravnej pozicie udajovej polozky vo vypise
$cislo_uctu_decoded = substr($cislo_uctu_partnera, 4, 5) .
    substr($cislo_uctu_partnera, 3, 1) . substr($cislo_uctu_partnera, 9, 1)
    . substr($cislo_uctu_klienta, 1, 2) . substr($cislo_uctu_partnera, 0,
    1); // zlozenie cifier cisla uctu do spravneho poradia
$predcislie_uctu_partnera = substr($lines[$i], 13, 6); // nacitanie
    predcislia uctu
$parsed_file['content'][$i]['account_number'] = $cislo_uctu_decoded; //
    priradenie cisla uctu do pola
$parsed_file['content'][$i]['account_prefix'] = $predcislie_uctu_partnera;
    // priradenie predcislia uctu do pola
```

4.5 Databázový model

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`.

Users

Druhou významnou tabuľkou sú 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.

Categories

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

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.

User_types

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

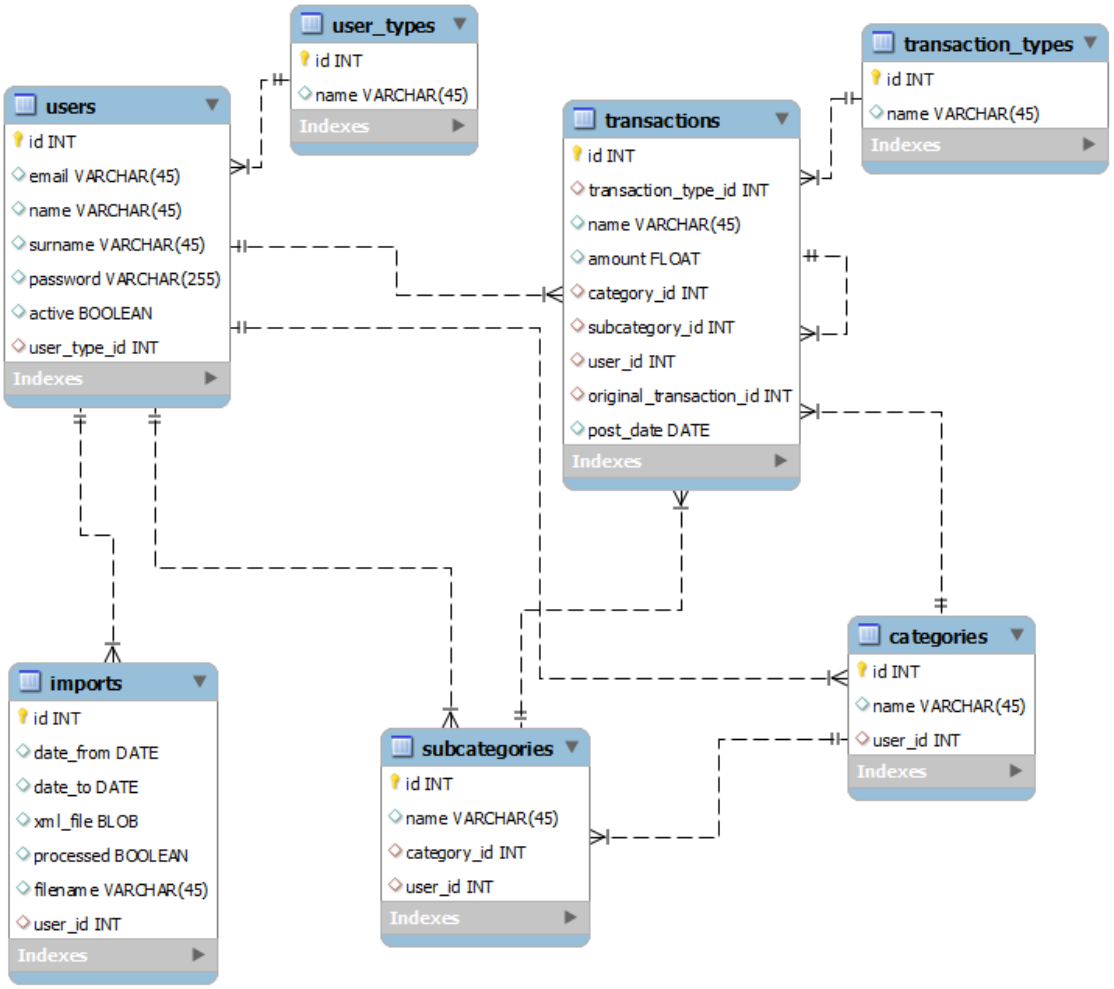
Transaction_types

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

Imports

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

4.6 Resetovanie hesiel



Obr. 4.1: Domáce účtovníctvo - Databázový model

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] The jQuery Foundation. jquery mobile, 2013.
http://jquerymobile.com/.

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

[20] W3C. HTML 5.1 nightly, 2013.
http://www.w3.org/html/wg/drafts/html/master/single-page.html
Zdroj navštívený: 12.5.2013.

Príloha

Priložené DVD médium

Priložené DVD obsahuje:

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