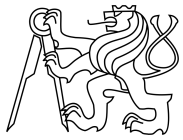
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Analyzátor finančních transakcí
Student:	Dávid Žalúdek
Vedoucí:	Ing. Jan Dufek
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2017/18

Pokyny pro vypracování

Cílem práce je vytvoření standalone aplikace, která umožní nahrát výpisy z bankovního účtu různých institucí a na jejich základech zobrazí zajímavé finanční statistiky či grafy. Uživatel bude mít možnost filtrovat vstupní data, nastavit časové období nebo granularitu. Aplikace bude vyvíjena jako open source v programovacím jazyce Java.
Součástí práce je také analýza typů statistik a grafů, které by mohly být pro uživatele zajímavé.

Metodika:

1) Proveďte analýzu typů statistik a grafů a vyhodnoťte, které by měly být použity.
2) Zhodnoťte a porovnejte možná existující řešení.
3) Proveďte analýzu možných technologií a vyberte nejvhodnější.
4) Udělejte návrh vzhledu a rozmístění ovládacích prvků.
5) Proveďte návrh implementace.
6) Implementujte návrh řešení.
7) Řešení otestujte na reálných uživatelích.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 10. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalárska práca

Analyzátor finančních transakcí

Dávid Žalúdek

Vedúci práce: Ing. Jan Dufek

30. decembra 2017

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 30. decembra 2017 .....

Odkaz na túto prácu

Žalúdek, Dávid. *Analyzátor finančních transakcí.* Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Práca sa zaoberá analýzou, návrhom, implementáciou a testovaním standalone aplikácie na spracovanie bankových výpisov. V práci sú tiež porovnané a zhodnotené už existujúce riešenia. Výsledkom práce je aplikácia s jednoduchým používateľským rozhraním, ktorá dokáže z nahraných dát vyprodukovať grafy. Táto aplikácia má nahrádzať aktuálne riešenia od bánk, ktoré sú zastaralé a neponúkajú túto funkcionalitu.

Klíčová slova java, grafy, grafické používateľské rozhranie, import dát

Abstract

Thesis deals with the problem of analysis, design and implementation of standalone application for analysis of banks statements. This work contains comparison and evaluation of existing solutions. The result of this work is application with simple user interface, which can create easily readable charts for end user. This application replaces current solutions mostly offered by banks which are dated and do not offer this functionality.

Keywords java, charts, grafical user interface, import of data

Obsah

Zoznam obrázkov

Úvod

V dnešnej dobe elektronického bankovníctva chýba užívateľovi možnosť nahliadnuť na svoje príjmy a výdaje v zrozumiteľnej forme. Banky ponúkajú rôzne možnosti stiahnutia výpisov z účtu, ale pre bežného užívateľa sú tieto informácie nečitateľné. Tento problém sa budem snažiť vyriešiť zobrazením týchto dát vo forme prehladných grafov.

Cieľ práce

Cieľom mojej bakalárskej práce je analýza, návrh a implementácia stand-alone aplikácie, ktorá bude slúžiť na analýzu výpisov z bankových účtov na základe zadaných vstupných parametrov. Súčasťou práce je tiež preštudovanie danej problematiky, zhodnotenie a porovnanie existujúcich riešení. Na záver je potrebné dokončenú aplikáciu otestovať a zhodnotiť navrhnuté riešenie.

Štruktúra práce

Táto práca je rozdelená do nasledujúcich kapitol:

- Existujúce riešenia - v tejto kapitole sa venujem analýze existujúcich riešení, ktoré ponúkajú funkcionalitu podobnú navrhovanej aplikácií.
- Analýza - táto kapitola popisuje požiadavky, ktoré sú kladené na vznikajúcu aplikáciu. Na to naväzuje popis prípadov použitia aplikácie.
- Návrh - táto časť práce obsahuje konceptuálny model, databázový model a návrh používateľského rozhrania.
- Realizácia - v tejto kapitole najskôr popíšem vybrané technológie a ich využitie v aplikácií. Následne načrtnem riešenia, ktoré som uplatnil v implementácií.

- Testovanie - v poslednej kapitole popisujem rôzne druhy testovania, ktoré som aplikoval pri vývoji výslednej aplikácie.

Kapitola 1

Existujúce riešenia

V tejto kapitole sa budem venovať rôznym aplikáciam na správu financií, z ktorých si môže užívateľ vyberať. Aplikácie popíšem a zhrniem výhody a nevýhody jednotlivých riešení.

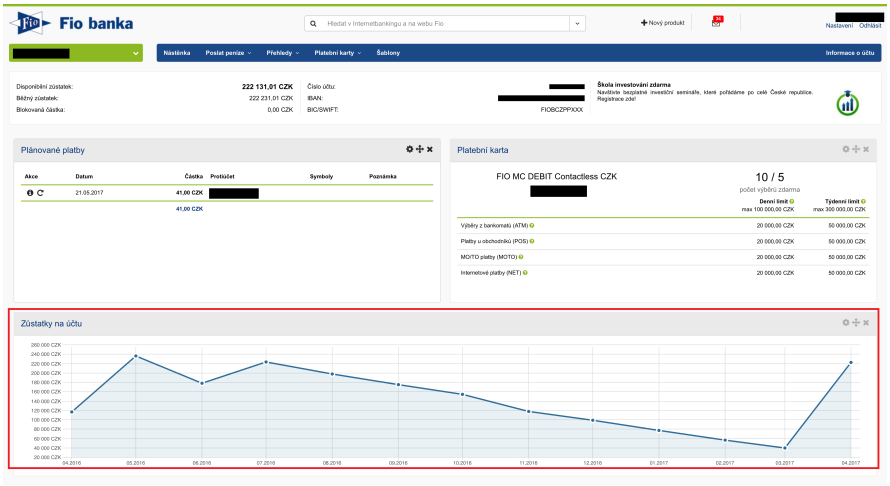
1.1 Bankové aplikácie

Sú to aplikácie vyvíjané bankami hlavne na správu účtov, ale ponúkajú aj iné funkcie. Napríklad export dát do strojovo čiteľného formátu alebo generovanie jednoduchých grafov ako je napríklad vývoj zostatku na účte. Pre užívateľa je tento spôsob spracovania dát najpríjemnejší lebo odpadá nutnosť inštalácie iných programov a importu dát.

+ Automatický import dát
+ Bezpečnosť
– Chýbajúca možnosť kombinovať výpisy z rôznych bánk
– Nedostatočná rôznorodosť grafov
– Nemožnosť kategorizovať dáta
– Nutné pripojenie k internetu

1.1.1 Fio banka

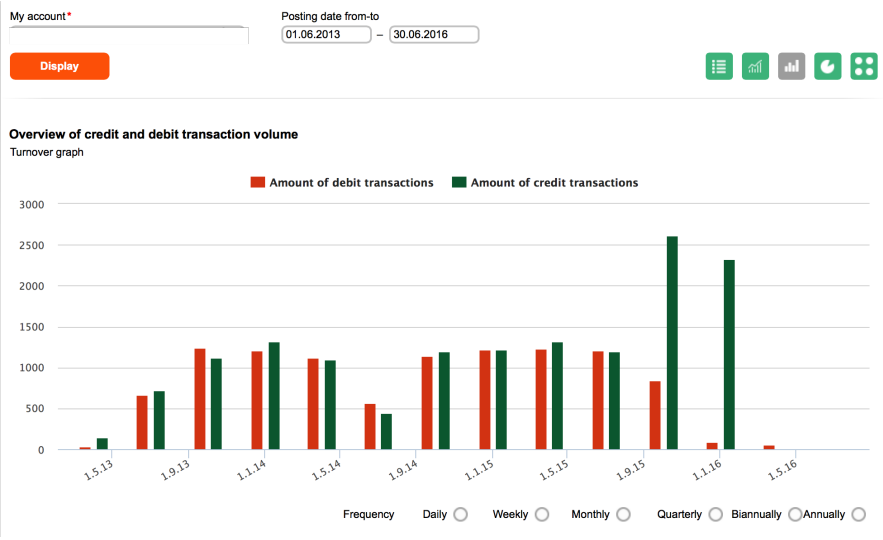
V mojej bakalárskej práci budem implementovať modul pre Fio banku. Táto banka ponúka vo svojom internetovom bankovníctve možnosť zobraziť graf vývoja zostatku na účte, v ktorom je možné nastaviť časový úsek a granularitu. (Obrázok 1.1)



Obr. 1.1: Webová aplikácia Fio banky

1.1.2 VÚB banka

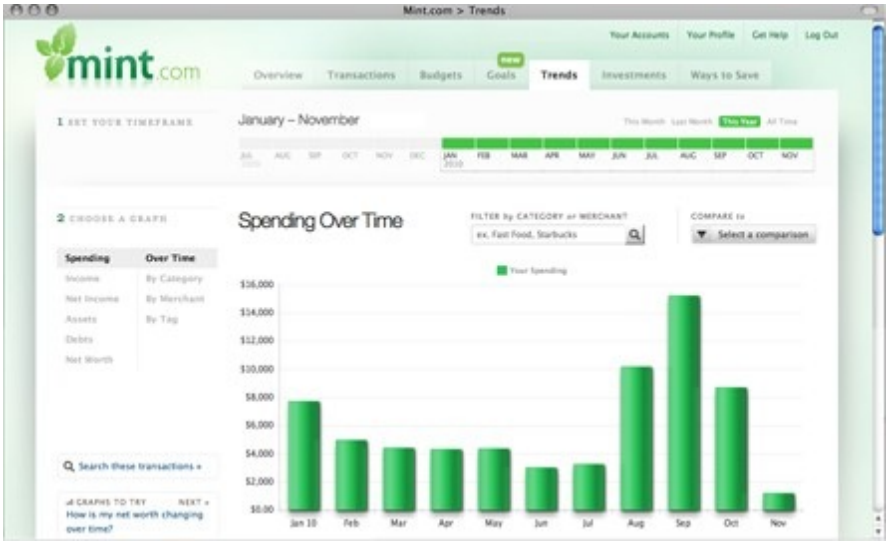
Vub.sk ponúka širšie možnosti reprezentácie dát. V ponúkaných grafoch nájdeme graf zobrazujúci zostatok na účte a stĺpcový graf, ktorý zobrazuje príjmy a výdaje. Pri oboch grafoch je možnosť nastaviť časové obdobie a granularitu (Obrázok 1.2). Táto banka ponúka aj možnosť zaradiť jednotlivé tranzakcie do kategórií, z ktorých následne generuje koláčovité grafy.



Obr. 1.2: Stĺpcový graf generovaný webovou aplikáciou VÚB banky

1.2 Webové aplikácie

Na trhu existuje veľa webových aplikácií, ktoré ponúkajú možnosť analyzovať dáta z bánk. Spomeniem napríklad `https://www.mint.com/` (Obrázok 1.3) alebo `https://pocketguard.com/`. Medzi ponúkané služby patrí napríklad vytváranie rozpočtu alebo automatická kategorizácia dát. Problémom týchto aplikacií je hlavne uzavretosť, čo je nevýhodou pre užívateľa z dôvodu analýzi jeho bankových tranzakcií poskytovaťelom aplikácie. Ďalšou nevýhodou je nedostatočná podpora bánk na európskom trhu.

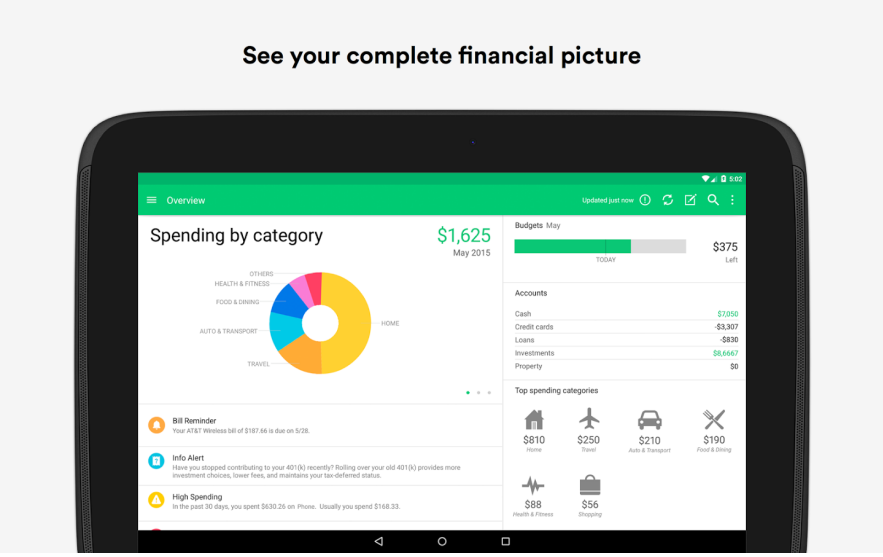


Obr. 1.3: Webová aplikácia mint.com

+ Veľký výber grafov a štatistík
+ Kategorizácia dát
+ Automatický import dát
– Nutné pripojenie k internetu
– Niektoré banky niesú podporované (chýba podpora CZ/SK bánk)
– Nie sú open-source

1.3 Mobilné aplikácie

Mobilné aplikácie sú vätčšinou zjednodušené webové aplikácie. Spomeniem napríklad You need a budget, Mint, Check - Bills and Money. Jednou z výhod je jednoduchosť používateľského rozhrania.



Obr. 1.4: Mobilná aplikácia mint.com

+ Jednoduché ovládanie
+ Kategorizácia dát
– Pripojenie k internetu
– Niektoré banky niesú podporované (chýba podpora CZ/SK bánk)
– Nie sú open-source

1.4 Šablóny

Bankové výpisy je možné spracovávať v štatisických programoch ako napríklad MS excel, Google Sheets alebo v štatistických jazykoch ako napríklad R alebo wolfram alpha. Toto riešenie vyžaduje znalosť jedného z prostredí a je cielené predovšetkým pre skúsenejších užívateľov.

+ Využitie pokročilejších štatistických metód
+ Open-source
– Nevhodné pre bežných užívateľov

1.5 Zhrnutie

Existuje veľa aplikacií na správu financií. U vätčšiny z nich sa nájdu nedostatoky, či už ide o uzavrenosť aplikácie, nedostatok funkcionality alebo neschopnosť analyzovať dáta z viacerých zdrojov. Preto vzniká potreba vyvinúť

aplikáciu, ktorá bude stand-alone a open-source s možnosťou rozširovania o ďalšie bankové inštitúcie.

Kapitola 2

Analýza

Analýza je jedna zo základných etáp pri vývoji softwaru. Je veľmi dôležitá, a preto je nutné ju detailne spracovať. Ide vlastne o popis toho, čo má daný software vykonávať. Kvalitnou analýzou sa predchádza chybám, ktoré sa môžu odhaliť až pri implementácií, a oprava je potom omnoho náročnejšia. Analýza softwaru väčšinou obsahuje súhrn funkčných a nefunkčných požiadaviek, diagramy prípadov použitia, diagramy aktivít popisujúce procesy u zákazníka a dátové modely.

2.1 Požiadavky

Táto podkapitola obsahuje popis všetkých požiadaviek, ktoré sú na vznikajúcu aplikáciu kladené. Požiadavky sú rozdelené do dvoch skupín: funkčné a nefunkčné. Funkčné požiadavky určujú chovanie a funkcie programu, nefunkčné špecifikujú vlastnosti a obmedzenia.

2.1.1 Funkčné požiadavky

F1 - Program bude schopný importovať dáta v strojovo čitateľnom formáte.

F2 - Program bude schopný importované dáta zpracovať, uložiť a znova načítať.

F3 - Program bude upozorňovať na duplikátne záznamy.

F4 - Program bude generovať grafy a štatistiky z nahratých dát.

F5 - Program bude kategorizovať záznamy.

F6 - Program bude spravovať viacero užívateľov.

Analýza požiadavku F1

Požiadavok F1 určuje, že môj program musí byť schopný importovať súbory v strojovo čitateľnom formáte. Najčastejším formátom výpisov z účtu s ktorou sa v bankovom sektore stretneme je formát CSV, no to nie je jediný formát ktorý by mala aplikácia podporovať. V následujúcej časti popíšem rôzne formáty a ich vlastnosti.

CSV - comma separated values Jedným z podporovaných formátov súboru pre import je CSV. CSV je acronymom pre "comma separated values", čiže hodnoty oddelené čiarkami. CSV je štandardizovaný normou RFC 4180 [1]. Výsadným problémom je norma, ktorá sa často nedodržiava kvôli zámene oddelovačov. S týmto formátom sa stretneme najčastejšie kvôli jednoduchému spracovaniu v každom tabuľkovom editore.

+ Šetrí dáta
− Nepodporuje dátovú hierarchiu
− Nepodporuje dátové typy
− Nepodporuje deserializáciu
− Nedodržovanie normy

XML - extensible markup language V XML je možné reprezentovať hierarchickú štruktúru dát, a základné dátove typy. Xml bol navrhnutý v roku 1996 a primutý ako W3C štandard v roku 1998 [2]. Tento formát nie je vo veľkej miere využívaný v bankovom sektore. Stavajú na ňom iné formáty, ako napríklad OFX, ktorý je štandardom pre export bankových výpisov do štatistických programov.

+ Možnosť reprezentovať hierarchickú štrukúru dát
+ Podpora deserializácie
+ Podpora dátových typov
− Veľkosť XML súboru

JSON - javascript object notation Vytvorený v roku 2001 ako náhrada za XML. Podobne ako XML je schopný reprezentovať hierarchickú štruktúru dát a dátové typy. JSON je štandardizovaný nomou RFC 7159 [3]. Tento formát je využívaný pri stahovaní dát z API bankovej inštitúcie.

+ Možnosť reprezentovať hierarchickú štrukúru dát

+ Podpora dátových typov
+ Podpora deserializácie

Iné

- OFX
  - Formát rozšírený hlavne v USA a Kanade. Export bankových dát do štatistických programov. [4]
- ABO
  - Formát používaný slovenskými a českými bankami na export a import tuzemských tranzakcií.

2.1.2 Analýza požiadavku F2

Tento funkčný požiadavok vznikol kvôli zjednodušeniu spravovania bankových výpisov. Uľahčí užívateľovi spravovať svoje bankové výpisy na jednom mieste. Vďaka tomuto požiadavku vzniká otázka zabezpečenia spravovaných dát.

2.1.3 Analýza požiadavku F3

Funkčný požiadavok číslo 3 definuje problém duplikátnych záznamov, ktoré sú nežiadúce pri zobrazovaní grafov. Predovšetkým kvôli reprezentácií skreslených a milných dát užívateľovy aplikácie. Najjednoduchším spôsobom zamedzenia duplikátov je porovnávanie všetkých pridaných záznamov z tými, ktoré sa už nachádzajú v databáze. Tento prístup nie je vyhovujúci, pretože aplikácia nemá slúžiť len na import z jedného účtu ale z viacerých, a takéto porovnanie by odfiltrovalo aj žiadúce záznamy. Preto som sa rozhodol nechať rozhodnúť o duplikátoch užívateľa. Užívateľovi budú prezentované záznamamy, ktoré sú vyhodnotené ako duplikáty a rozhodne o ich ponechaní alebo zmazaní.

2.1.4 Analýza požiadavku F4

Aplikácia bude generovat niekolko typov grafov a štatistík, ktoré budú zobrazované užívateľovi v interaktívnej podobe podľa nastaviteľného filtra. Detailnejšie tento problém rozoberiem v nasledujucej podkapitole.

2.1.5 Analýza požiadavku F5

Tento funkčný požiadavok vnikol kvôli sprehladneniu dát. Program ponúkne užívateľovi možnosť priradiť jednu z kategórií záznamu a potom program priradí automaticky rovnakú kategóriu všetkým podobmým záznamom. Toto riešenie ulahčí kategorizáciu a umožní vytváranie zaujímavejších štatistík a grafov.

2.1.6 Nefunkčné požiadavky

N1 - Ľahká rozšíriteľnosť.

N2 - Stand-alone applikácia v Jave.

N3 - Intuitívne ovládanie.

N4 - Validácia vstupov.

Analýza požiadavku N1

Vzhľadom na to že banky nemajú jednotný formát poskytovania dát, vzniká nutnosť navrhnúť aplikáciu tak, aby bola možnosť pridávať banky ako moduly do existujúcej aplikácie. Takisto ako možnosť pridávať dalšie grafy a štatistiky do aplikácie.

Analýza požiadavku N2

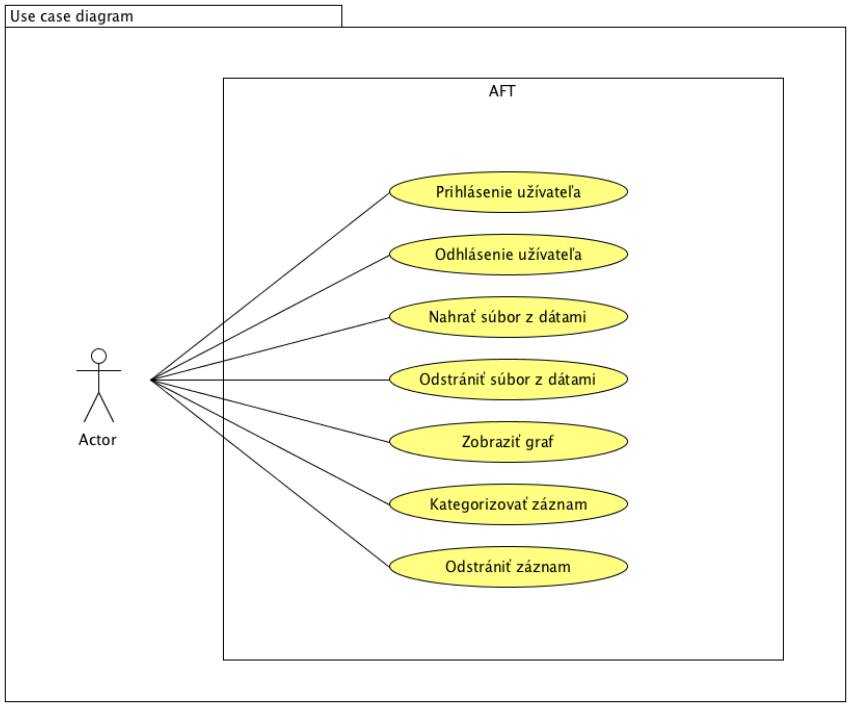
Aplikácia bude vyvíjaná v programovacom jazyku Java z dôvodu multiplatformosti. Aplikácia bude stand-alone vzhľadom na to, že bude spravovať citlivé užívateľské dáta.

Analýza požiadavku N3

Aplikácia bude mať intuitívne ovládanie a bude jednoducho použiteľná aj pre neskúsených používateľov. Grafické rozhranie bude prehľadné a responzívne.

2.2 Prípady použitia

Model prípadov použitia definuje funkcie, ktoré budú dostupné užívateľovi. Vychádza z funkčných požiadaviek, ktoré sú následne detailne rozobraté do prípadov použitia. Tiež popisuje rôznych používateľov systému.



Obr. 2.1: Diagram prípadov použitia programu

2.2.1 Popisy prípadov použitia

1. Prihlásenie užívateľa
   - Užívateľ zadá meno a heslo do formulára, aplikácia overí správnosť a nahrá jeho uložené dáta.
2. Odhlásenie užívateľa
   - Aplikácia zmaže všetky užívateľské dáta z dočastnej pameti.
3. Nahrať súbor s dátami
   - Užívateľ zvolí súbor, ktorý sa má nahrať do aplikácie a zadá meno banky do formulára.
4. Odstránenie súboru s dátami
   - Užívateľ zvolí súbor, ktorý sa má odstrániť z aplikácie.
5. Zobrazenie grafu
   - Užívateľ zvolí typ grafu a vyplní formulár, na základe ktorého bude vygenerovaný graf z nahratých dát.

6. Odstránenie záznamu zo súboru
    - Užívateľ zvolí súbor a následne zvolí záznam, ktorý sa má odstrániť.
7. Kategorizovanie záznamu
    - Užívatel zvolí kategóriu alebo vytvorí novú pre záznam.

2.2.2 Zoznam účastníkov

Na zozname účastníkov sú dvaja užívateľia, ktorí sa od seba odlišujú len prihlásením. Užívateľ importuje nové dáta, pokiaľ je prihlasený, dáta sa ukladajú a je možné ich znova načítať.

2.2.3 Scenáre prípadov použitia

Scenáre prípadov použitia slúžia na popis jednotlivých krokov, ktoré sa budú vykonávať v konkrétnych prípadoch použitia.

Hlavný scenár použitia - zobrazenie grafu

1. Scenár prípadu použitia začína, keď používateľ dostane nový výpis z banky.
2. Používateľ sa prihlási do aplikácie.
3. Po prihlásení sa zobrazí formulár na import súboru. Tu užívateľ zvolí meno banky zo zoznamu podporovaných bánk a súbor zo systému.
4. Aplikácia následne spracuje súbor a vyhodnotí či sa v nom nenachádzaju duplikátne záznamy. Ak áno, aplikácia navrhne užívateľovi odstránenie týchto záznamov.
5. Užívateľ potom zobrazí okno z grafmi, kde vyberie jeden z ponúkaných grafov.
6. Užívateľ následne môže nastavovať rôzne vlastnosti grafu, ako napríklad časové obdobie alebo granularitu.
7. Scenár prípadu použitia končí odhlásením užívateľa.

Vedlajší scenár použitia - kategorizácia záznamov

1. Scenár prípadu použitia začína, keď sa používateľ rozhodne pre detalnejšiu analýzu dát.
2. Používateľ sa prihlási do aplikácie.
3. Po prihlásení si rozklikne záznamy a pridelí záznamom kategórie.

4. Užívateľ potom zobrazí okno z grafmi, kde vyberie jeden z ponúkaných grafov.

5. Užívateľ následne môže nastavovať rôzne vlastnosti grafu, ako napríklad časové obdobie alebo granularitu.

6. Scenár prípadu použitia končí odhlásením užívateľa.

2.3 Analýza bankových výpisov

V tejto sekcii sa budem venovať dátam, ktoré banky ponúkajú užívateľovi na stiahnutie. A rozoberiem do detailov dáta poskytované bankami, ktoré budem implemntovať ako moduly do mojej aplikácie.

2.3.1 Obsah výpisov z účtu

Jedná sa o zoznam transakcií na danom účte, kde od každej transakcie očakávame že bude obsahovať položky: Čas, Suma, Typ tranzakcie, Mena. Ale môže obsahovať aj zaujímavejšie dáta ako napríklad poloha, kategória transakcie alebo názov spoločnosti, ktorá prijala platbu. Tieto extra dáta sú veľmi zaujímave, ale neposkytuje ich každá banka a nie sú štandardizované. Z toho vyplýva, že musia byť spracované samostatne pre každú banku.

Fio banka

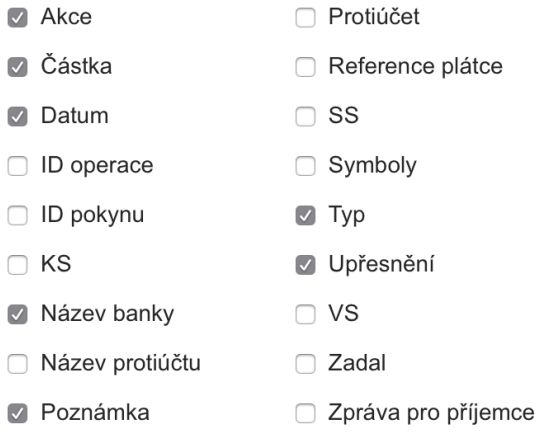
Fio banka ponúka veľa možností prístupu k dátam. Všetky sú nadefinované v API [5]. Okrem štandardných položiek, táto banka ponúka aj polohu, kde bola transakcia vykonaná a názov spracovávateľa transakcie. Poznámka má síce nedefinovaný formát v API štandarde, ale z pozorovania výpisov som zistil, že z nej nebude problém vytiahnuť aspon meno spoločnosti, ktoré pomôže z automatickou kategorizáciou. Po zvolení časového úseku a položiek 2.2 zo zoznamu, ktoré má exportovaný súbor obsahovať, sú ponúknuté 4 možnosti formátu:

- CSV
  - Tu je možnosť voliť zo zoznamu položiek 2.2, ktoré má výsledný CSV súbor obsahovať. Preto nemôžme očakávať jednotnú štruktúru dát.
- CSV API
  - Táto možnosť úplne ignoruje užívateľovu voľbu položiek a dodržuje štruktúru, ktorá je jasne definovaná v API.
- OFX

  - Tento formát je asi jeden z najzaujímavejších, ktorý banka ponúka. Riadi sa štandardom a je jednoducho deserializovateľný. [4]

- GPC

  - Jedná sa o formát, ktorý je nadefinovaný v API [5]. Problémom je zložitá deserializácia.



Obr. 2.2: Formulár exportovaných položiek Fio.cz

Vúb banka

Vúb banka ponúka rozsiahle možnosti filtrácie transakcií 2.3, ako aj široké množstvo formátov [6]:

- ABO

  - Jedná sa o formát ktorý je nadefinovaný v API [6]. Nevýhodou je zložitá deserializácia.

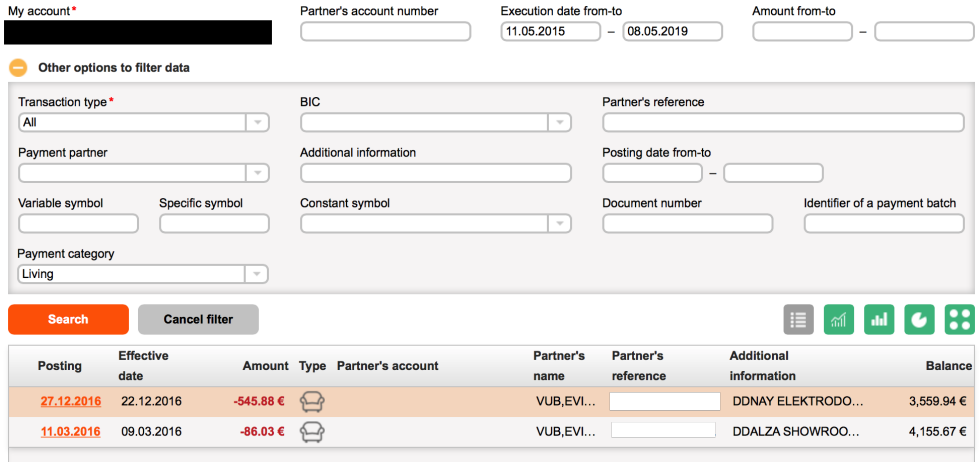
- CSV

  - Zvolený formát, kvôli jednoduchosti importu dát.

- XML

- XLS

  - Tabuľka vo formáte vhodnom na spracovanie v Microsoft Excel.

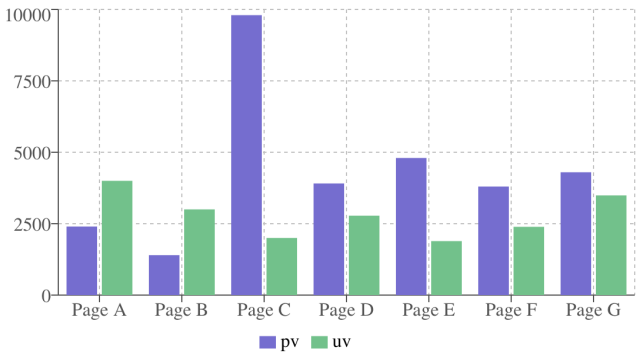


Obr. 2.3: Formulár exportu Vub.sk

2.4 Analýza typov grafov a štatistík

V tejto kapitole sa budem venovať jednotlivým grafom, ktoré sú zaujímavé pre užívaťela a rozoberiem štatistiky, ktoré budem z dát vytvárať. Pri analýze využitia grafov som sa riadil doporučeniami z Charts and Graphs for Microsoft Office Excel 2007 [7] kde autor popisuje rôzne využitia grafov.

Stĺpcový graf



Obr. 2.4: Stĺpcový graf

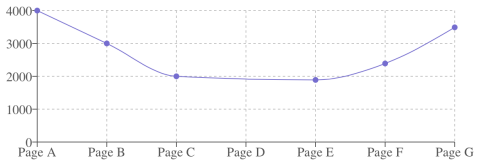
Stĺpcový graf zobrazuje dáta za použitia horizontálnych alebo vertikálnych stĺpcov, ktoré porovnávajú diskrétne veličiny. Jedna os zobrazuje jednotlivé porovnávané kategórie, druhá hodnotu diskrétnej veličiny. Tento typ grafu je používaný hlavne na zobrazenie kategorických dát. Kategorické dáta sú dáta

rozdelené do diskrétnych skupín ako napríklad mesiace v roku, dni v týždni, veková skupina a iné.

Využitia v rámci mojej aplikácie :

- Výdaje v časovom úseku (mesiace, dni, roky)
  - Jedná sa o súčet všetkých tranzakcií, ktoré boli vykonané počas definovaného časového úseku. Užívateľovi tento graf má načrtnúť trend výdajov na účte.

Čiarový graf

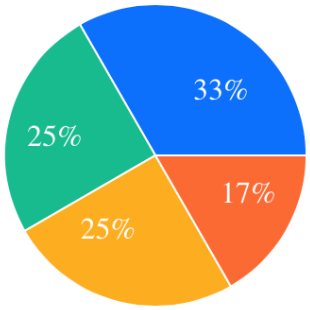


Obr. 2.5: Čiarový graf

Tento graf zobrazuje informáciu ako sériu bodov spojených čiarov. Body sú najčastejšie v časovom poradí. Využíva sa na zobrazenie trendu v dátach počas určitého časového obdobia. Tento graf nebudem používať na zobrazovanie zostatku na účte, z dôvodu neexistencie záznamu o štartovnej bilancií v bankových dátach. Do tohto grafu bude zakompovaná možnosť preložiť graf krivkou, ktorá pomôže užívateľovi zobraziť trend výdajov alebo príjmov.

- Akumulované výdaje
  - Body grafu reprezentujú súčet výdajov do daného dátumu, z tohto grafu sú jasne viditeľné zmeny vo výdajoch na účte s postupom času.
- Akumulované príjmy.
  - Body grafu reprezentujú súčet užívateľových príjmov do daného dátumu.
- Vývoj zostatku na účte.
  - Tento graf zobrazuje krivku vývoja stavu účtu počas daného obdobia.

Koláčový graf



Obr. 2.6: Koláčový graf

Jedná sa o kruh rozdelený do niekoľkých sekcií, ktoré zobrazujú percentuálny podiel jednotlivých veličín. V aplikácií bude zobrazovať pomery výdajov/príjmov v rôznych kategóriách:

- Typy tranzakcií.
- Banky.
- Kategórie zaznamov.
- Názvy spoločností.
-

Aritmetický priemer

Jedná sa o súčet prvkov v postupnosti vydelený počtom členov.

Vzorec: $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$.

- Priemerná útrata za časové obdobie.
- Priemerná útrata za jedlo, oblečenie alebo inú kategóriu v časovom období.

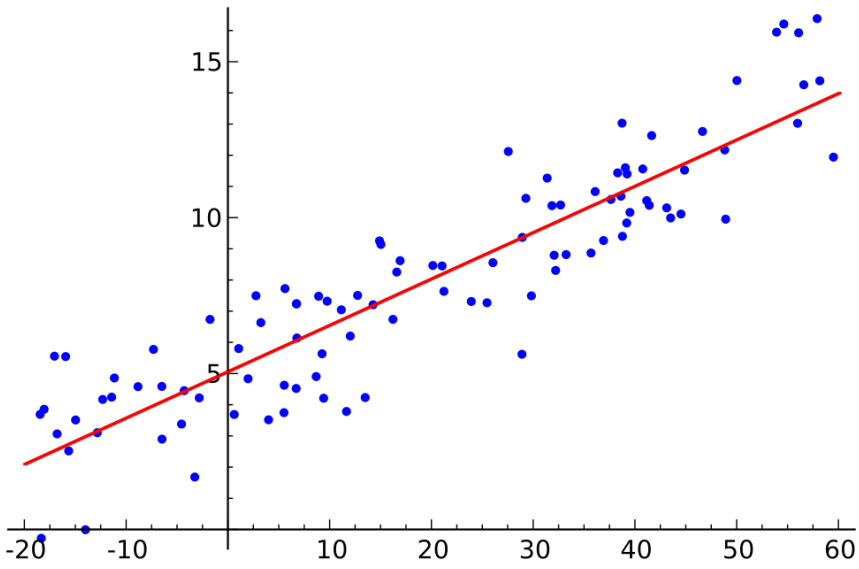
Súčet

Vzorec: $y = \sum_{i=1}^{n} x_i$

- Celková útrata za deň, mesiac, rok.
- Celkový príjem/útrata v banke.
- Celková útrata v kategórii.

Lineárna regresia

V jednoduchej lineárnej regresii sa pokúšame modelovať vzťah medzi dvoma veličinami. Napríklad príjmom a počtom rokov vzdelania, výškou a hmotnosťou ľudí, nadmorskou výškou a bodom varu vody alebo dávkou lieku a reakciou ľudského tela. Slúži na to, aby si užívateľ vedel predstaviť celkový trend výdajov a príjmov, a mohol ich k vzájomne porovnať. [8]



Obr. 2.7: Lineárna regresia

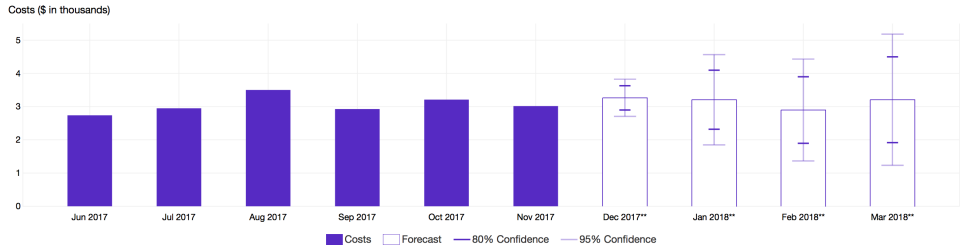
Výpočet:
Pri výpočte sa snažíme nájsť priamku $f(x) = ax + b$, ktorá najlepšie aproximuje vzťah medzi danými veličinami. K odhadu a a b použijeme metódu minimálnych štvorcov.

$$a = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2}$$

$$b = \frac{\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2}$$

Konfidenčný interval

Konfidenčný interval je typ intervalového odhadu, ktorý vychádza z pozorovaných dát. V konfidenčnom intervale urččujeme mieru vierohodnosti. Najčastejšie sa používa 95% konfidenčný interval, na základe ktorého sme schopní vypočítať horný a dolný interval s pomocou ktorých vieme predpovedať kam budú spadať s 95% pravdepodobnosťou nasledujúce hodnoty v postupnosti. Túto štatistiku využívam pri stĺpcovom grafe kde spočítam konfidenčný interval pre nasledujúce týždne, mesiace alebo roky.



Obr. 2.8: Konfidenčný interval

Pri výpočte konfidenčných intervalov je nutné si uvedomiť o aké rozdelenie sa jedná. Pri vybere rozdelenia mi pomohla publikacia The Statistics of Payments [9] ktorá traznakcie modeluje pomocov log-normálového rozdelenia. Po analýze histogramu načítanej vzorky tranzakcií som sa taktiež rozhdol pre log-normalové rozdelenie. Na výpočet konfidenčného intervalu log-normálového rozdelenia som použil metódu popisanu v 3.6 Confidence Intervals for the Mean of a Log-Normal Distribution [10], takže som rozdelenie aproximoval normalovým rozdelením.

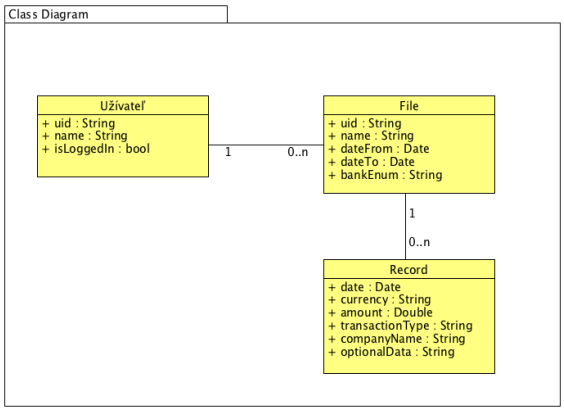
Kapitola 3

Návrh

Návrh je dôležitou fázou pri tvorbe softwaru. Vychádza z analýzy funkčných a nefunkčných požiadaviek, na základe ktorých sa následne vytvorí konceptualny model, databázový model, ako aj návrch používateľského rozhrania. Do návrhu tiež spadá výber návrhových vzorov a ich zakomponovanie do aplikácie.

3.1 Konceptuálny model

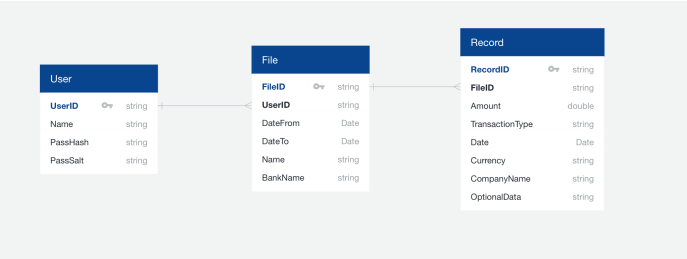
Konceptuálny model je jednou z prvých fáz dátového modelovania. Vymedzujú sa v ňom všetky informácie, ktoré sú pre aplikáciu relevantné a treba si ich uschovávať. Konceptuálny model mojej aplikácie je zobrazený na obrázku 3.1. Hlavným prvkom aplikácie je užívateľ, korý môže mať niekoľko súborov, ktoré obsahujú v sebe záznamy o jednotlivych transakciách.



Obr. 3.1: Konceptuálny model

3.2 Databázový model

Databázový model zobrazuje dáta programu, ktoré sa budú uschovávať a ich náväznosť. Na obrázku 3.2 je zobrazený logický model datábazy. Databázu som navrhoval z dôrazom na jednoduchú rozšíriteľnosť o ďalšie dáta špecifické pre banku. Na túto schopnosť slúži pole optionalData v tabuľke Records.



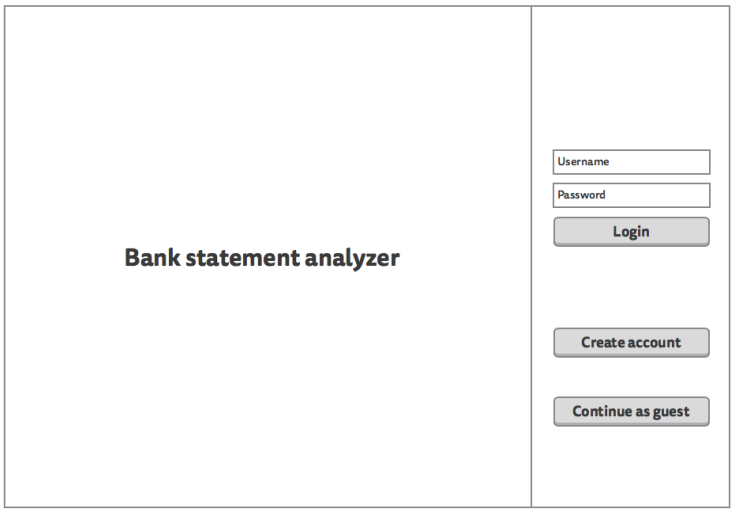
Obr. 3.2: Logický model databázy

3.3 Návrh používateľského rozhrania

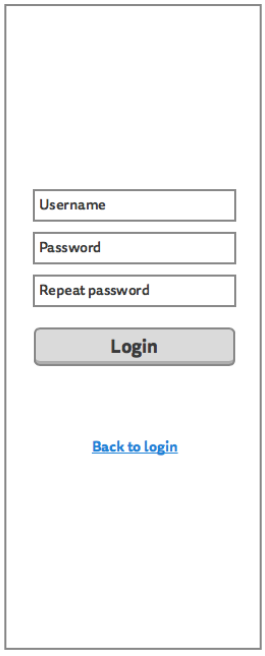
Pri návrhu aplikácie je dôležité dbať na prehľadnosť a jednoduché používanie. Ešte pred samotnou impementáciou aplikácie, je dobré zamyslieť sa nad rozložením komponentov užívateľského prostredia. Na toto mi poslúžili wireframe diagramy.

3.3.1 Prihlasovacie rozhranie

Prvá vec s ktorou príde užívateľ do kontaktu po spustení aplikácie, je prihlasovacia stránka 3.3. Na ktorej sú mu ponúknuté možnosti: prihlásiť sa, pokračovať ako hosť alebo vytvoriť nový účet. Na vytvorenie nového účtu slúži formulár.



Obr. 3.3: Wireframe prihlasovacej stránky



Obr. 3.4: Wireframe registrovacieho formulára

3.3.2 Štruktúra užívatelského rozhrania

Aplikácia sa bude skladať z 2 hlavných častí:

- Hlavička 3.5

- Obsah

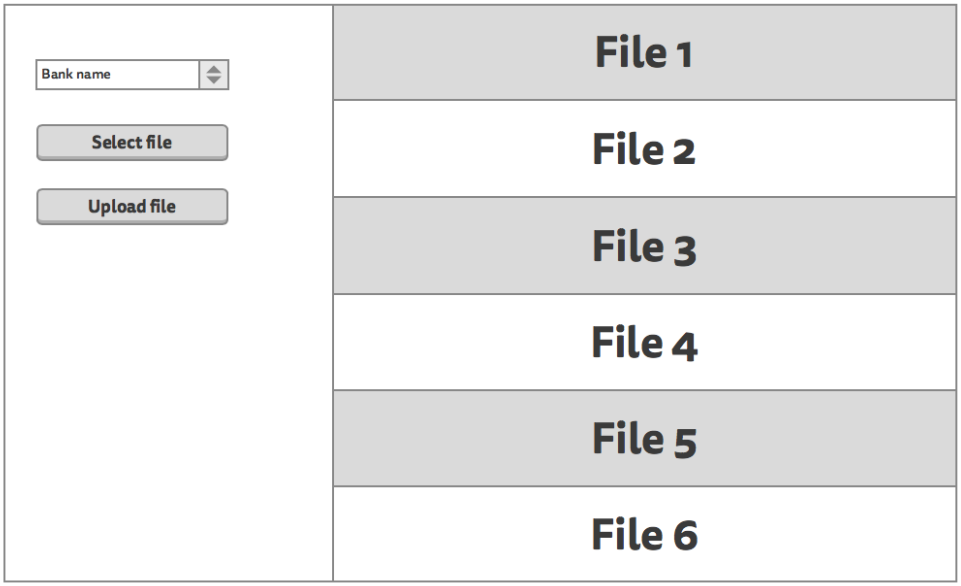
3.3.3 Hlavička

Hlavička naľavo obsahuje tlačítko domov, ktoré užívatelovi zobrazí domovskú stranku, tlačítko files zobrazí tabuľku s aktuálne načitanými súbormi, tlačítko grafs zobrazí rozhranie na výber grafu a tlačítko records zobrazí tabuľku načítaných záznamov o tranzakciách. Na pravo sa uživatelovi zobrazuje meno prihláseného účtu a tlačítko log out, ktoré slúži na odhlásenie užívatela a návrat na prihlasovaciu stránku.

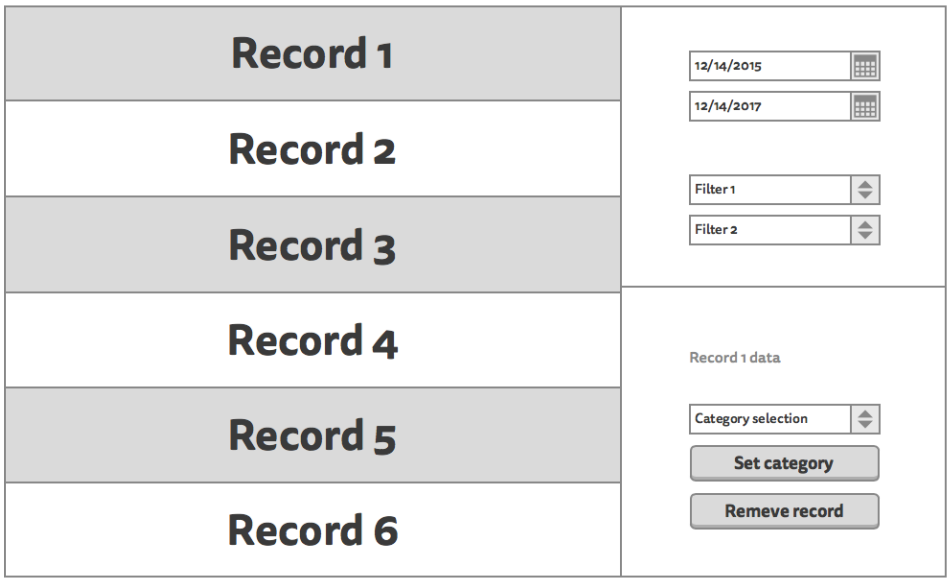
Obr. 3.5: Wireframe hlavičky

3.3.4 Obsah

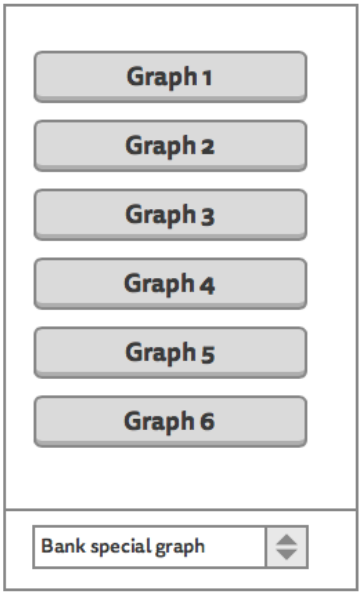
- Tabuľka súborov 3.6
  - Tabuľka súborov sa skladá z formulára na nahranie súboru a zoznamu súborov.
- Tabuľka záznamov 3.7
  - Tabuľka záznamov sa skladá z listu záznamov, ktorý je filtrovatelný pomocou filtrov v pravom paneli. V poslednej rade sa tu nachádza panel, ktorý zobrazi všetky dáta v zázname a umožňuje prideliť kategóriu.
- Výber grafov 3.6
  - Sklada sa z
- Zobrazenie vybraného grafu 3.9
  - Zobrazenie sa skladá z grafu a jeho príslušného filtru.



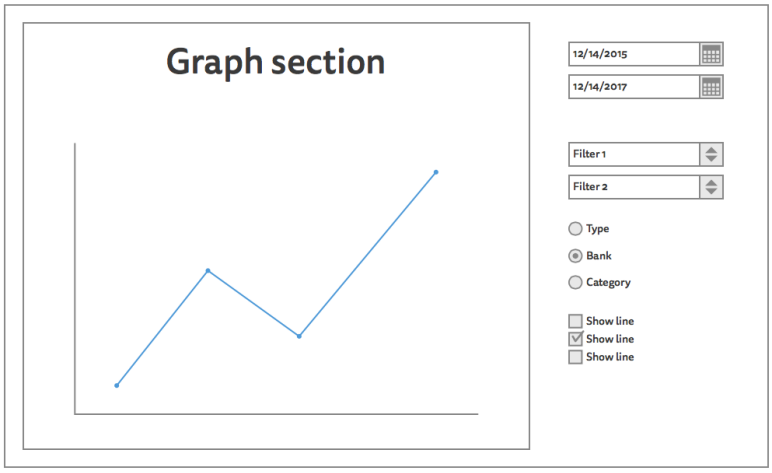
Obr. 3.6: Wireframe zobrazenia tabuľky so súbormi



Obr. 3.7: Wireframe zobrazenia záznamov



Obr. 3.8: Wireframe výberu grafov



Obr. 3.9: Wireframe zobrazenia grafu

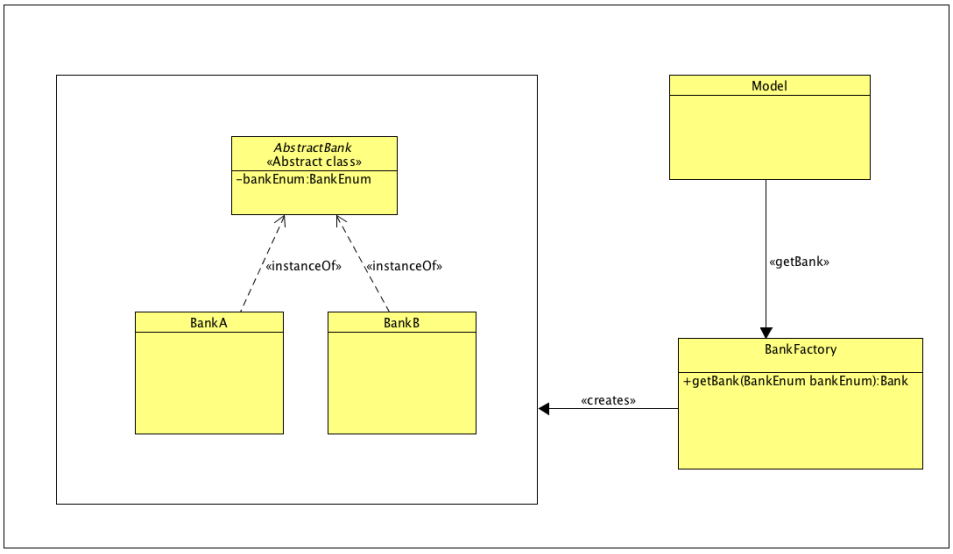
3.4 Návrhové vzory

Aplikácia bude využívat návrhové vzory, ktoré riešia opakované problemy pri návrhu softwaru. Medzi najznámejšiu literatúru z tejto problematiky patri Design patterns [11] inak pomenovaný Gang of Four (GoF). Návrhové vzory sa rozdelujú do 3 kategorií:

- Vzory vytváracie
  - Do týchto vzorov patria štruktúry, ktoré zabezpečujú správne vytváranie, inicializáciu a počet vytváraných objektov.
- Vzory štrukturálne
  - Štrukturálne vzory slúžia na sprehladnenie a usporiadanie vytváraných tried alebo komponent aplikácie.
- Vzory chovania
  - Riešia problémy dedičnosti a správania jednotlivych tried alebo komponent.

3.4.1 Factory

Patrí medzi vytváracie vzory. Vytvára konkrétnu implementáciu abstraktnej triedy alebo rozhrania. Tento návrhový vzor bude zakomponovaný do mojej aplikácie pri vytváraní konkrétnych implementacií bánk a grafov.

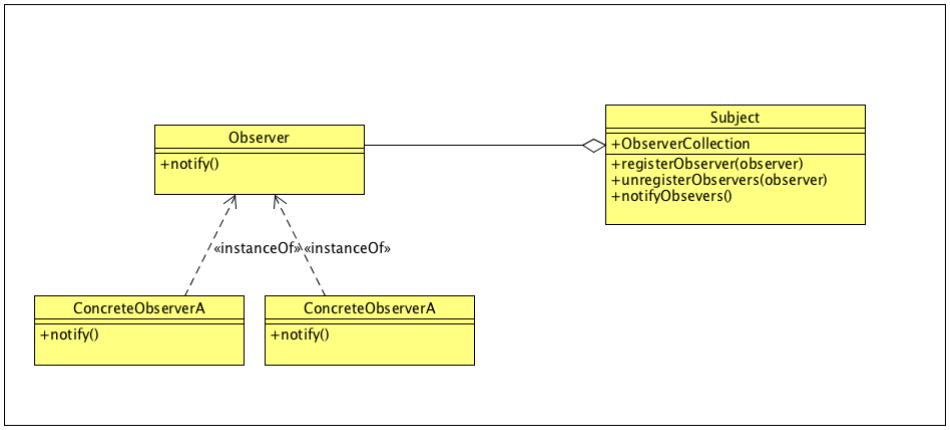


Obr. 3.10: Návrhový vzor BankFactory

3.4.2 Observer

Patrí medzi vzory chovania. Zabezpečuje predávanie dát medzi vláknami, používa k tomu vydavatela a predplatitela. Vydavatel informuje predplatitelov o zmene stavu. V mojej aplikácií bude tento vzor zahrnutý do vzťahu modelu

a užívatelského rozhrania, kde jednotlivé triedy budú informované o zmene stavu, na základe ktorého upravia svoje komponenty.



Obr. 3.11: Návrhový vzor observer

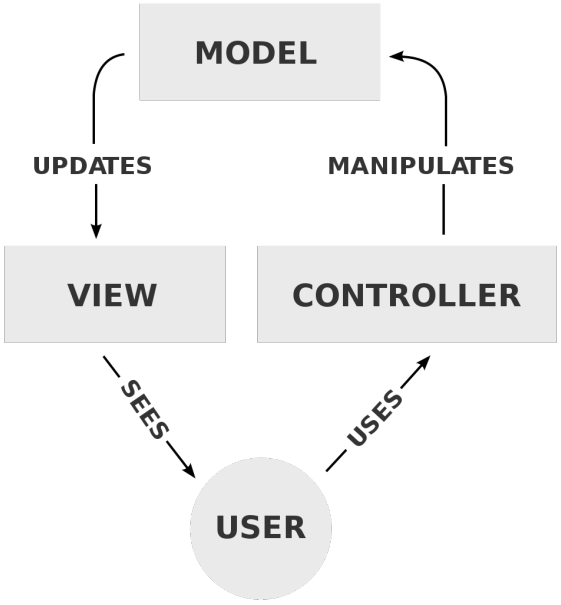
3.5 Navrh architektúry aplikácie

V tejto časti sa budem zaoberať návrhom štruktúr, ktoré budú zabezpečovať splnie všetkých funkčných a nefunkčných požiadaviek. Ako základ aplikácie som si zvolil model-view-controller, ktorý je štandardom pri vytváraní aplikácií z užívatelským rozhraním.

3.5.0.1 Model view controller

MVC 3.12 sa skladá z 3 častí:

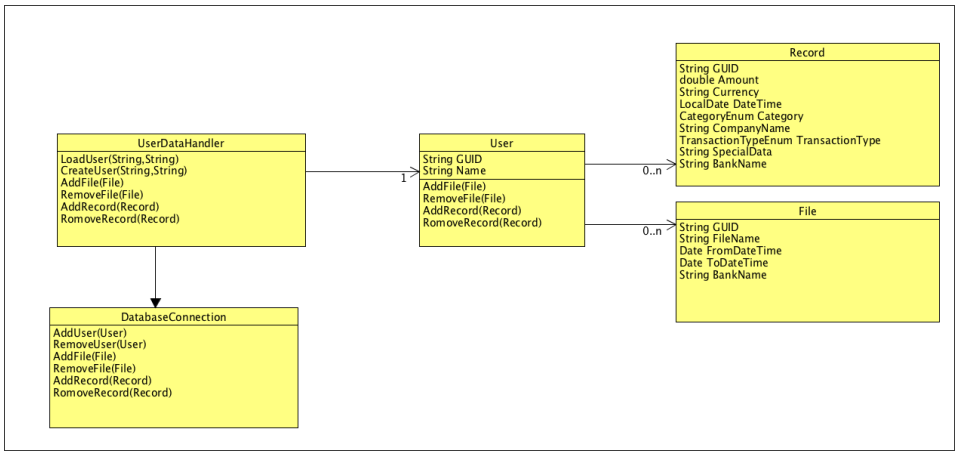
- Model
  - Je centrálnou komponentou návrhového vzoru. Stará sa o logickú a dátovu vrstvu aplikácie. Je nezávislý od implementácie užívatelského rozhrania.
- View
  - Interpretuje uživatelovi dáta z modelu. Môže existovať mnoho rôznych implementácií view naväzujúcich na rovnaký model.
- Controller
  - Spracováva vstupy od užívatela, na základe ktorých sa upravuje model.



Obr. 3.12: Model view controller diagram

3.5.0.2 Návrh modelu

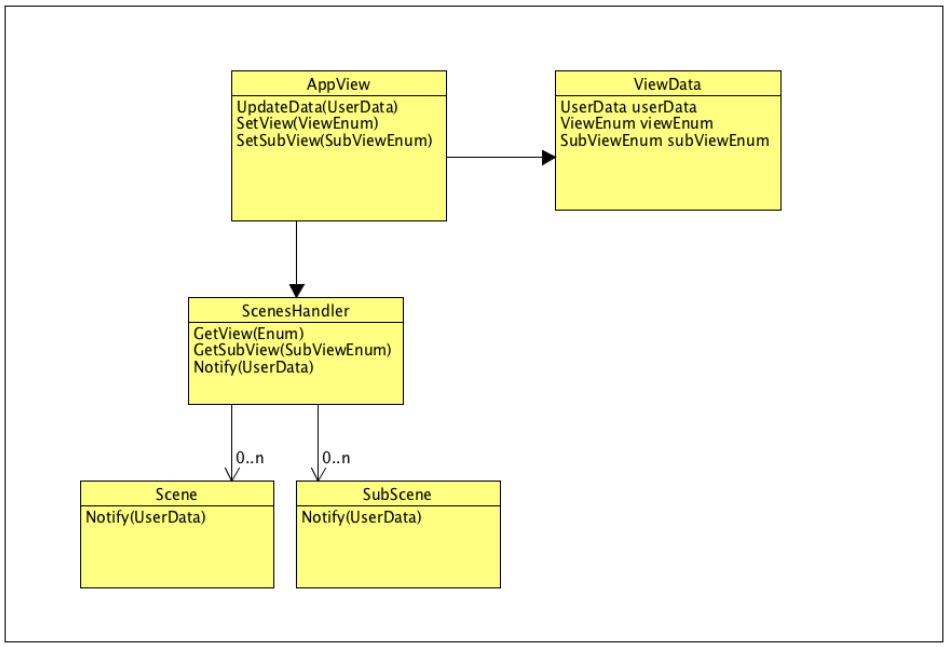
Hlavnou komponentou modelu je UserDataHandler, ktorý Spravuje všetky užívatelské dáta a zaisťuje synchronizáciu s databázou. Pri každej zmene užívatelských dát sa aktualizuje view.



Obr. 3.13: Diagram modelu

3.5.0.3 Návrh view

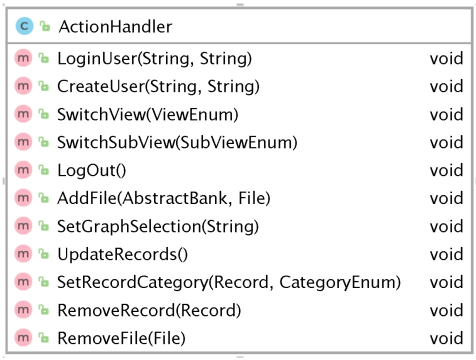
Pri návrhu view som použil návrhový vzor observer. Vždy keď sú užívatelské dáta aktualizované v AppView, tak sa zavolá notify na všetkých objektoch view, vďaka čomu sa aktualizuje užívatelské rozhranie. View je navrhnutý tak, aby nezasahoval priamo do modelu, ale aby všetky užívatelské vstupy delegoval controlleru.



Obr. 3.14: Diagram view

3.5.0.4 Návrh controlleru

Controller je navrhnutý ako jednoduchý objekt ActionHandler, ktorý spravuje užívatelské vstupy. Následne ich predáva modelu na spracovanie alebo rovno nastavuje scénu vo view.



Obr. 3.15: Diagram controlleru

Kapitola 4

Realizácia

V tejto kapitole priblížim realizáciu navrhovanej aplikácie, problémy, ktoré som riešil, a nakoniec zhrniem do akého štádia sa mi podarilo aplikáciu implementovať.

4.1 Vybrané technológie

Podľa nefunkčných požiadaviek z analýzy, navrhovaná aplikácia má byť v programovacom jazyku java, pre používateľa jednoducho použiteľná, ľahko rozšíritelná s intuitívnym ovládaním. Týmito kritériami som sa riadil pri výbere technológií.

4.1.1 Databáza

Ako databázu som sa rozhodol použiť SQLite [12]. Jedná sa o relačnú databázu, ktorej hlavnými prednosťami je jednoduchosť nasadenia, kompaktnosť balíka, podpora JDBC [13] a rýchlosť vývoja. Jednou z nevýhod SQLite je nepodporovatelnosť šifrovania dát rôznych užívatelov. SQLlite je šírená pod Public Domain takže spĺňa licenčné kritéria na použitie v mojej praci.

4.1.2 Užívatelské rozhranie

Užívatelské rozhranie je implementované pomocou knižnice JavaFX [14], ktorá je štandardom pri vývoji desktopových aplikácií v jazyku java. Ponúka široké množstvo užívatelských prvkov, štandardné grafy, jednoduché spracovanie udolastí a možnosť použiť CSS na úpravu vzhľadu konečného užívatelského rozhrania. Knižnica je licencovaná pod BCL, ktorá umožňuje distribúciu binárnych súborov bez úprav.

4.1.3 Matematická knižnica

V analýze popisujem lineárnu regresiu a konfidenčný interval. O ich výpočet sa bude starať knižnica Apache Commons Math [15], ktorá obsahuje všetky potrebné štatistické metódy. Knižnica je licencovaná pod Apache Licence [16].

4.1.4 Parsovacie knižnice

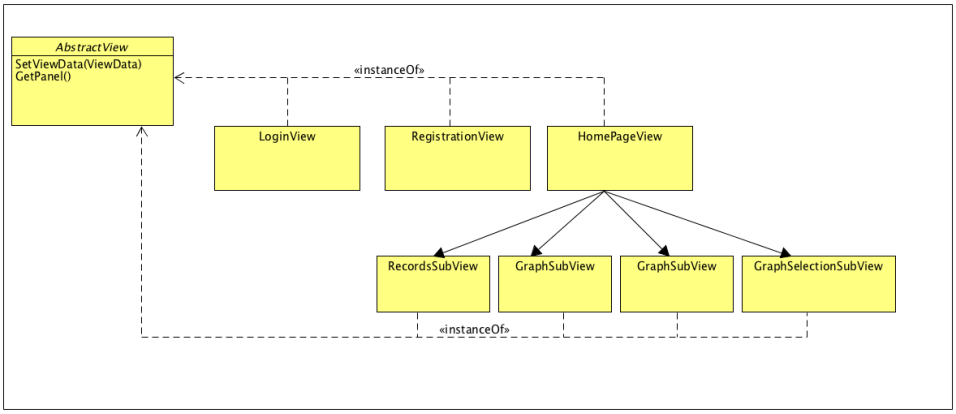
Na parsovanie CSV súborov som vybral knižnicu OpenCSV.

4.2 Implementácia

Implementácia je predposledné štádium vývoja softvéru. Vychádza z návrhu a analýzy. V tejto časti popíšem implementáciu jednotlivých častí aplikácie.

4.2.1 Tvorba užívatelského rozhrania

Užívatelské rozhranie bolo vytvorené v programe SceneBuilder, ktorého výstupom sú fxml súbory. Tieto súbory definujú prvky a ich rozlozenie v scéne. Pri tvorbe užívatelského rozhrania som sa riadil wireframami z návrhu. Prepínanie medzi jednotlivými scénami som zabezpečil pomocou jednej hlavnej scény, ktorá bola zložená z hlavičky a obsahu. Obsah sa vykresloval na základe aktuálne nadefinovaného SubViewEnum. Výsledný strom jednotlivých scén je zobrazený v obrázku C.5.



Obr. 4.1: Diagram implementácie užívatelského rozhrania

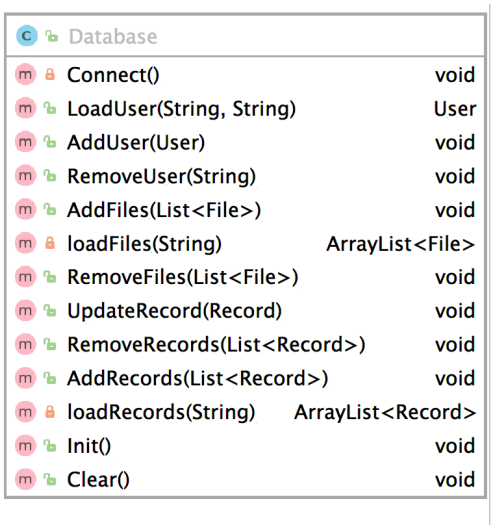
4.2.2 Správa užívatelských dát

Užívatelské dáta sú v mojej aplikácií spravované objektom UserDataHandler, ktorý zabezpečuje synchronizáciu dát medzi aplikáciou a databázou. Je to

jediný objekt, ktorý spravuje užívatelské dáta, čím je zaručená ich stabilita. Užívatelsk

4.2.3 Implementácia databázy

Databázu som implementoval podľa relačného návrhu. Do databáze sa pristupuje z objektu DatabaseHandler, ktorý implementuje rozhranie na prístup ku dátam. Použitá databáza je kompatibilná s JDBC štandardom, takže je možnosť SQLite zameniť za inú z kompatibilných databáz. Pri implementácií databázy som naprogramoval aj vlastné výnimky: UserNotFoundException a UserExistsException, ktoré sú vyhadzované pri načítaní a odstraňovaní užívatela z databázy.



Obr. 4.2: Diagram triedy DatabaseHandler

4.2.4 Validácia dát

Validáciu dát je zabezpečovaná vo viacerých vrstvách aplikácie. Prvá validácia prebieha pri nahrávaní súboru. Aplikácia kontroluje, že sa jedná o príponu súboru podporovanou instanciou nahrávanej banky. Ďalšia validácia prebiaha vo vnútri parsovacej funkcie. Tu sa vyradia záznamy, ktoré sú nekompletné alebo poškodené. Posledná kontrola prebieha v UserDataHandler, ktorý na základe vygenerovaných hashov zo záznamov vyradí tie, ktoré sa už nachádzajú v zozname.

4.2.5 Implementácia bankových modulov

Na vytváranie bankových modulov používam BankFactory, ktorá zabezpečuje načítanie bankových instancií a udržiava ich zoznam. Bankové moduly sú

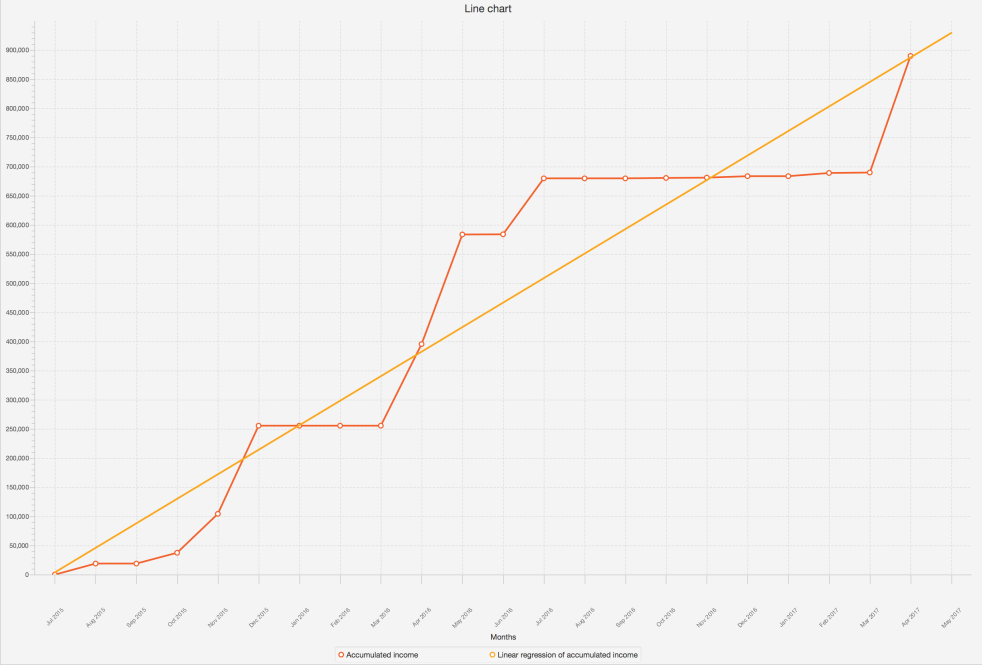
do aplikácie načítavané pomocou ClassLoaderu, ktorý počas behu programu načíta imlementácie AbstractBank a uloží ich do listu. Výhodou riešenia je možnosť pridávať banky ako class súbory umiestnené v adresári banks bez nutnosti upravovať vnútorné prvky.

4.2.6 Implementácia grafových modulov

FilterFactory zabezpečuje vytváranie instancií AbstractFilter. Táto továrňa taktiež používa ClassLoader. Instancie sa načítajú z adresára filters, no obsahuje do seba zakomponované aj štandardné grafy, ktoré su generované len z povinných polí záznamu a súboru.

4.2.6.1 Čiarový graf

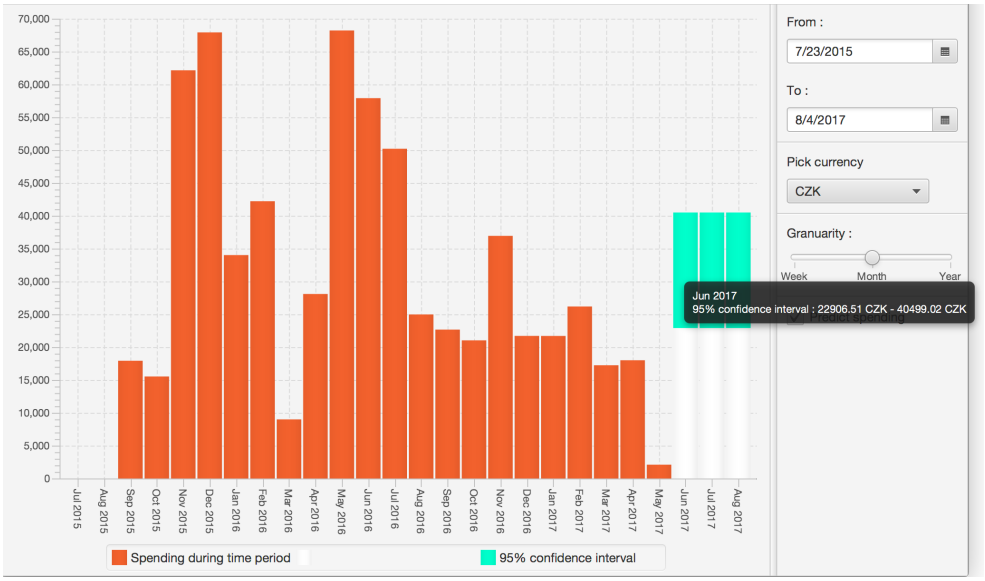
Jedná sa o implementáciu čiarového grafu ako AbstractFilter do mojej aplikácie. Je to jeden zo štandardných grafov, ktoré má užívatel k dizpozícií pre všetky bankové subjekty. Pri vytváraní tohto grafu som sa riadil analýzou a zakomponoval som doňho údaje o akumulované príjmy a výdaje.



Obr. 4.3: Čiarový graf s lineárnou regresiou

4.2.6.2 Stĺpcový graf

Štandardný filter stĺpcového grafu generuje graf, ktorý ponúka možnosť pužiť konfidenčné intervaly na predpoveď nasledujúcich mesiacov a týždňov.



Obr. 4.4: Stĺpcový graf z konfidenčným odhadom

4.2.7 Automatická kategorizácia

Automatickú kategorizáciu som implementoval ako mapu, kde mapujem názov spoločnosti na jednu z preddefinvaných kategórií. Perzistenciu riešim serializáciou mapy do súboru pri každej zmene mapy.

4.2.8 Zhrnutie

Aplikácia je implementovaná podľa návrhu. Pri programovaní som sa nestretol z vážnejším problémom. Myslím že sa o to zaslúžil hlavne dobrý návrh a výber knižníc, kroré majú dobrú dokumentáciu.

Kapitola 5

Testovanie

Testovanie je fáza vývoja, ktorá sa vykonáva počas celého procesu implementácie aplikácie. Slúži na odhalenie chýb softvéru.

5.1 Kontrola kódu

Jedná sa o jeden zo statických testov. Má za úlohu odhaliť logické chyby v kóde. Odporúča sa aby kontrolu vykonával iný programátor, ako autor kódu. Toto testovanie som využíval hlavne pri vývoji. Doplnil som to unit testami, ktoré som písal len na vybrané moduly mojej aplikácie.

5.2 Statická analýza kódu

Pri statickej analýze kódu sa hladajú chyby v návrhových vzoroch, ako napríklad zachytávanie výnimok alebo nedosažitelnosť niektorých z vetiev. Vzhľadom na to, že som pri vývoji používal vývojové prostredie Intelij idea, ktoré túto analýzu vykonáva automaticky, nemusel som statickú analýzu kódu vykonávať ručne.

5.3 Unit testy

Unit testy slúžia na testovanie jednotlivých modulov aplikácie. Ich hlavná výhoda je možnosť automatizácie. Pri mojej implementácií som si najskôr nadefinoval rozhranie daného modulu, a potom voči nemu napísal test. Na tvorbu unit testov som používal knižnicu Junit 5 [17], vybral som ju kvôli jednoduchej podpore v použitom IDE. Unit testy som vytvoril pre jednotlivé bankové moduly, databázu a pre category parser.

5.4 Testovanie nefunkčných požiadaviek

V týchto testoch som hlavne overoval funkčnosť aplikácie na najpoužívanejších operačných systémoch. Tieto testy odhalili chybu pri načítavaní suboru v bankových moduloch. Pri bežnom načítaní súboru do streamu sa použilo defaultné kódovanie systému, čo bolo nežiadúce.

5.5 Funkčné a systémové testy

Tieto testy sú vykonávané na konci vývoja. Funkčné testy zisťujú, či program zodpovedá funkčným požiadavkám. U systémových testov sa zisťuje funkčnosť systému ako celku. Aplikáciu som testoval, či splňuje všetky funkčné požiadavky zo sekcie 2.1.1.

5.6 Testovanie na užívateloch

Aplikáciu som testoval na skupine známich. Väčšine sa aplikácia páčila. Výhrady boli hlavne na rýchlosť vstupu do aplikácie, ktoré mohlo trvať niekoľko sekúnd. Tento problém som vyriešil aktualizovaním len scény, ktorá sa zobrazuje užívatelovi. Toto riešenie zrýchlilo celkovú responzívnosť aplikácie.

5.7 Zhrnutie

Testy boli veľkov pomocou pri realizácií a záverečnom hodnotení funkčnosti. Aplikácia spĺňa všetky funkčné a nefunkčné požiadavky. Týmto je fáza testovania ukončená.

Záver

Cieľom tejto bakalárskej práce bolo navrhnúť a implementovať program na štatistické spracovanie bankových výpisov. Podľa zadania som zanalyzoval existujúce riešenia, identifikoval som nedostatky, ktoré som následne odstránil v návrhu. V návrhu som sa venoval navrhu databázy, vytvoril wireframy a špecifikoval architektúru aplikácie. Následne som v realizácií popísal ako prebiehala implementácia a riešenia, ktoré som zakomponoval do aplikácie. Nakoniec som popísal použité testovacie metódy a vytvorenú aplikáciu otestoval na užívateloch.

Výsledkom tejto bakalárskej práce je stand-alone aplikácia v programovacom jazyku java, ktorá spracováva bankové výpisi, umožňuje užívatelovi kategorizovať a zprehladniť bankové dáta. Týmto programom bol splnený cieľ práce.

Aplikácia je aktuálne pripravená na rozšírenie pomocou nových bánkových a grafových modulov. Ďalším smerom, ktorým sa vývoj aplikácie môze uberať je pridanie sofistikovanejšej metódy kategorizovania dát, napríklad formou umelej inteligencie, ktorá na kategorizáciu dát používa širšie spektrum dátových vstupov, nie len meno spoločnosti, ale napriklad aj sumu a dátum tranzakcie.

Literatúra

[1] Common Format and MIME Type for Comma-Separated Values (CSV) Files. October 2005, [cit. 2017-5-10]. Dostupné z: https://www.ietf.org/rfc/rfc4180.txt

[2] Extensible Markup Language (XML) 1.0 (Fifth Edition). November 2008, [cit. 2017-5-10]. Dostupné z: https://www.w3.org/TR/REC-xml/

[3] The JavaScript Object Notation (JSON) Data Interchange Format. March 2014, [cit. 2017-5-10]. Dostupné z: https://tools.ietf.org/html/rfc7159

[4] Open Financial Exchange. [online], [cit. 2017-05-12]. Dostupné z: http://www.ofx.net

[5] Fio banka: API Bankovnictví. [online], [cit. 2017-05-12]. Dostupné z: https://www.fio.cz/docs/cz/API_Bankovnictvi.pdf

[6] Vúb banka: Popis štruktúry technických formátov exportných súborov. [online], 2016, [cit. 2017-05-12]. Dostupné z: https://www.vub.sk/files/osobne-financie/ucty-platby/nonstop-banking/navody-manualy/formaty-vypisov_export.pdf

[7] Jelen, B.: *Charts and Graphs for Microsoft Office Excel 2007*. Que Publishing, 2007, ISBN 0789736101.

[8] Rencher, A. C.; Schaalje, G. B.: *Linear Models in Statistics*. Wiley-Interscience, 2008, ISBN 0471754986.

[9] A BILLION HERE, A BILLION THERE: THE STATISTICS OF PAYMENTS. [cit. 2017-5-10]. Dostupné z: https://swiftinstitute.org/wp-content/uploads/2012/10/The-Statistics-of-Payments_v15.pdf

[10] Confidence Intervals for the Mean of a Log-Normal Distribution. [cit. 2017-5-10].

[11] Gamma, E.: *Design patterns : elements of reusable object-oriented software*. Reading, Mass: Addison-Wesley, 1995, ISBN 0-201-63361-2.

[12] SQLite. [online], [cit. 2017-12-15]. Dostupné z: `https://www.sqlite.org`

[13] Oracle: Java JDBC API. [online], [cit. 2017-12-15]. Dostupné z: `https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/`

[14] Oracle: JavaFX. [online], [cit. 2017-12-15]. Dostupné z: `https://docs.oracle.com/javafx/2/`

[15] Apache Commons Math. [online], [cit. 2017-12-15]. Dostupné z: `http://commons.apache.org/proper/commons-math/`

[16] Apache License, Version 2.0. [online], [cit. 2017-12-15]. Dostupné z: `http://www.apache.org/licenses/LICENSE-2.0`

[17] JUnit 5. [online], [cit. 2017-12-15]. Dostupné z: `http://junit.org/junit5/`

Dodatok **A**

Zoznam použitých skratiek

GUI Graphical user interface

XML Extensible markup language

CSV Comma-separated values

XLS Spreadsheet (Microsoft Excel) file format

API Application programming interface

OFX Open Financial Exchange

CSS Cascading Style Sheets

RFC Request for Comments

MVC Model view controller

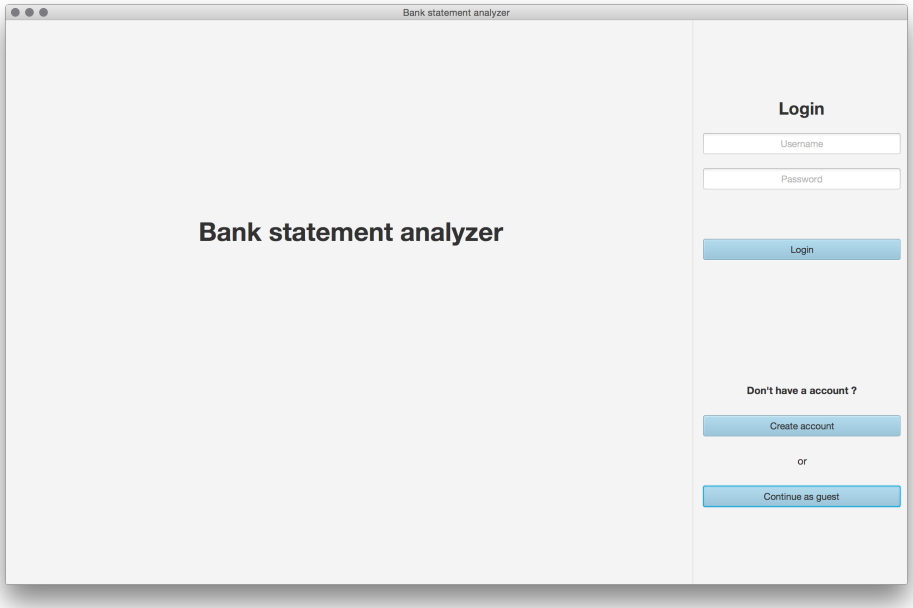
Dodatok B

Obsah priloženého CD

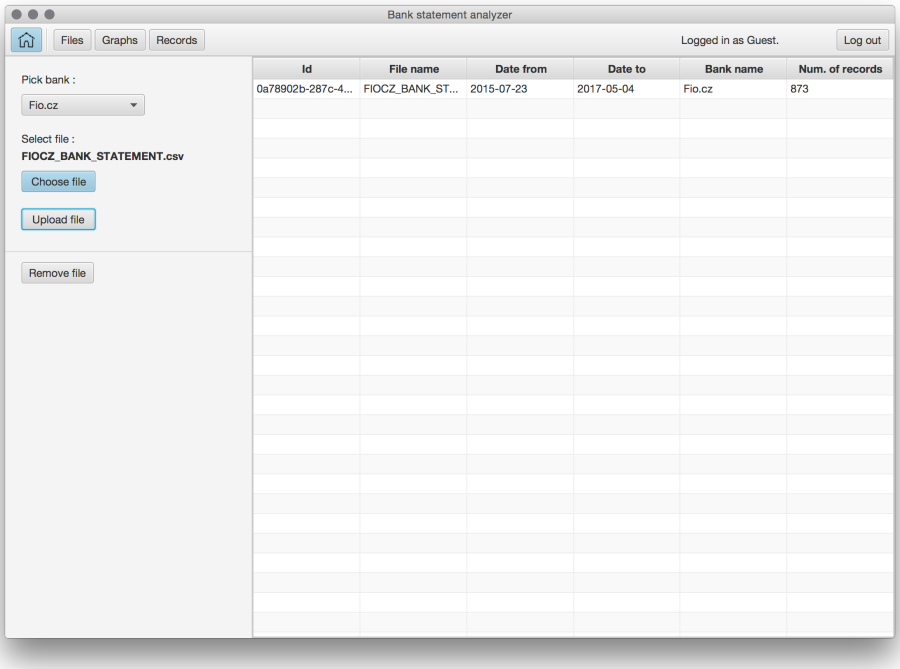
- `readme.txt` ..... stručný popis obsahu CD
- `bin` ..... adresár so spustiteľnou formou implementácie
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF

Dodatok C

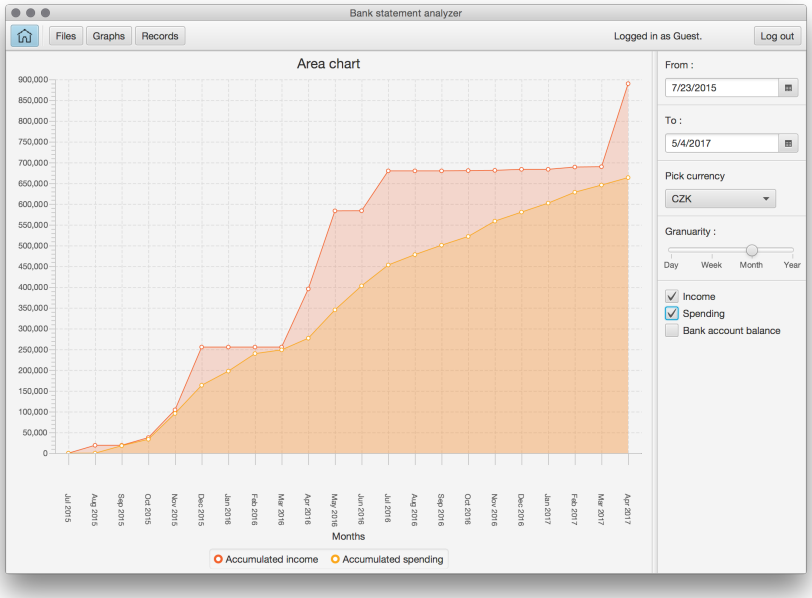
Snímky aplikácie



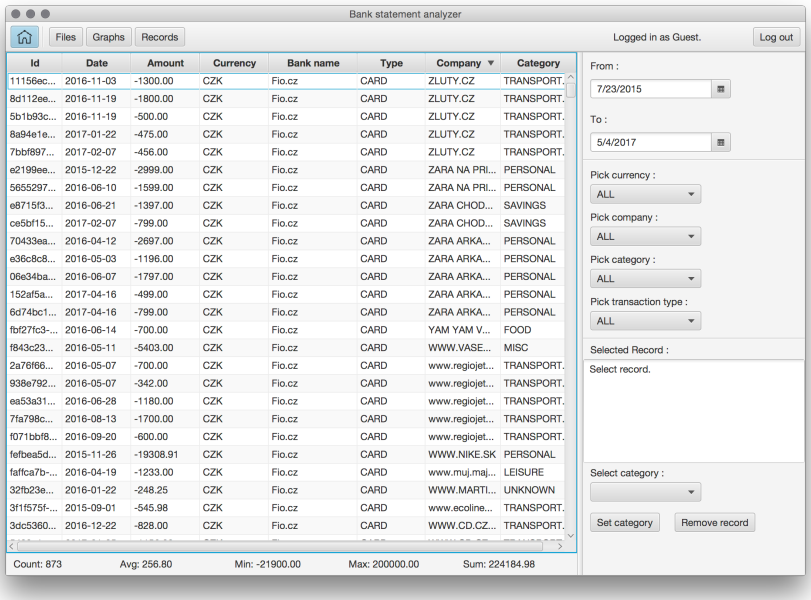
Obr. C.1: Login stránka



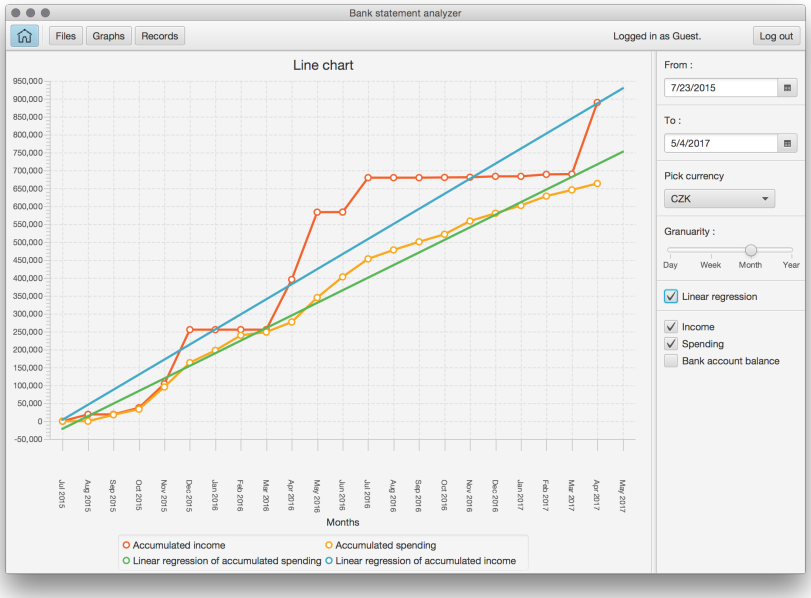
Obr. C.2: Zoznam súborov



Obr. C.3: Zobrazenie plošného grafu



Obr. C.4: Zobrazenie záznamov



Obr. C.5: Zobrazenie čiarového grafu