

Sem vložte zadanie Vašej práce.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
KATEDRA SOFTWAREVÉHO INŽENÝRSTVÍ



Bakalárska práca

Analyzátor finančních transakcí

Dávid Žalúdek

Vedúci práce: Ing. Jan Dufek

18. mája 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 18. mája 2017

.....

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2017 Dávid Žalúdek. Všetky práva vyhrazené.

Táto práca vznikla ako školské dielo na FIT ČVUT v Prahe. Práca je chránená medzinárodnými predpismi a zmluvami o autorskom práve a právach súvisiacich s autorským právom. Na jej využitie, s výnimkou bezplatných zákonných licencií, je nutný súhlas autora.

Odkaz na túto prácu

Žalúdek, Dávid. *Analýzátor 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 funkcionality.

Kľúčová slova webová aplikácia, 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 web application, graphical user interface, import of data

Obsah

Úvod	1
1 Existujúce riešenia	3
1.1 Bankové aplikácie	3
1.2 Webové aplikácie	5
1.3 Mobilné aplikácie	5
1.4 Šablóny	6
1.5 Zhrnutie	6
2 Analýza	9
2.1 Požiadavky	9
2.2 Prípady použitia	12
2.3 Analýza bankových výpisov	14
2.4 Analýza typov grafov	17
3 Návrh	21
3.1 Konceptuálny model	21
3.2 Databázový model	22
3.3 Návrh používateľského rozhrania	22
4 Realizácia	25
4.1 Vybrané technológie	25
4.2 Zvolené knižnice	25
4.3 Tvorba používateľského rozhrania	27
4.4 Správa užívateľských dát	29
4.5 Validácia dát	29
4.6 Konverzia kurzov	29
4.7 Vykreslenie grafu	29
4.8 Implementácia bankových modulov	30
4.9 Zhrnutie	30

5	Testovanie	33
5.1	Kontrola kódu	33
5.2	Statická analýza kódu	33
5.3	Testovanie nefunkčných požiadaviek	33
5.4	Funkčné a systémové testy	33
	Záver	35
	Literatúra	37
A	Zoznam použitých skratiek	39
B	Obsah priloženého CD	41

Zoznam obrázkov

1.1	Fio.cz internet banking	4
1.2	Vub.sk graf	4
1.3	Webová aplikácia mint.com	5
1.4	Mobilná aplikácia mint.com	6
2.1	Diagram prípadou použitia	13
2.2	Tabuľka položiek vo výpise Fio.cz	15
2.3	Formulár exportu Vub.sk	16
2.4	Stĺpcový graf	17
2.5	Čiarový graf	18
2.6	Koláčový graf	18
3.1	Konceptuálny model	21
3.2	Databázový model	22
3.3	Wireframe hlavičky	22
3.4	Wireframe zobrazenia grafu	23
3.5	Wireframe zobrazenia súborov	23
4.1	Redux štruktúra	26
4.2	Responzivnosť aplikácie	28

Úvod

V dnešnej dobe elektronického bankovníctva chýba užívateľovi možnosť nahliadnuť na svoje príjmy a výdaje v jednoducho zrozumiteľnej forme. Banky ponúkajú možnosti stiahnutia výpisov z účtu, ale pre bežného užívateľa sú tieto informácie nečitateľné. Tento problém sa budem snažiť vyriešiť zobrazením týchto dát vo forme jednoduchých 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 zadáných vstupných parametrov. Súčasťou práce je tiež preštudovanie danej problematiky a 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 funkcionality podobnú navrhovanej aplikácii.
- Analýza - táto kapitola popisuje požiadavky, ktoré sú kladené na vznikajúcu aplikáciu. Na to naväzuje popis prípadou použitia aplikácie.
- Návrh - táto kapitola obsahuje konceptuálny model, databázový model a návrh používateľského rozhrania.
- Realizácia - v tejto kapitole najskôr popíšem vybrané technológie a ich využitie v aplikácii, a následne načrtnem riešenia ktoré som uplatnil v implementácii.

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

Existujúce riešenia

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

1.1 Bankové aplikácie

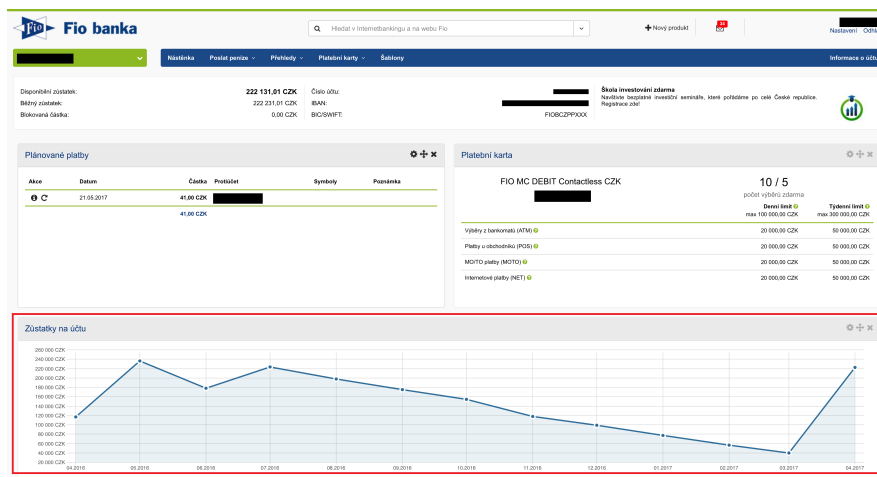
Aplikácie vyvíjané bankami hlavne na správu účtu, ale ponúkajú aj iné funkcie, a to napríklad export dát do strojovo čitateľného formátu alebo generovanie jednoduchých grafov ako je napríklad vývoj zostatku na účte. Pre užívateľa je tento spôsob spracovania dát asi najpríjemnejší lebo odpadá nutnosť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

- + Automatický import dát
- + Bezpečnosť
- Nemožnosť kombinovať výpisy z rôznych bánk
- Nedostatok možností 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)

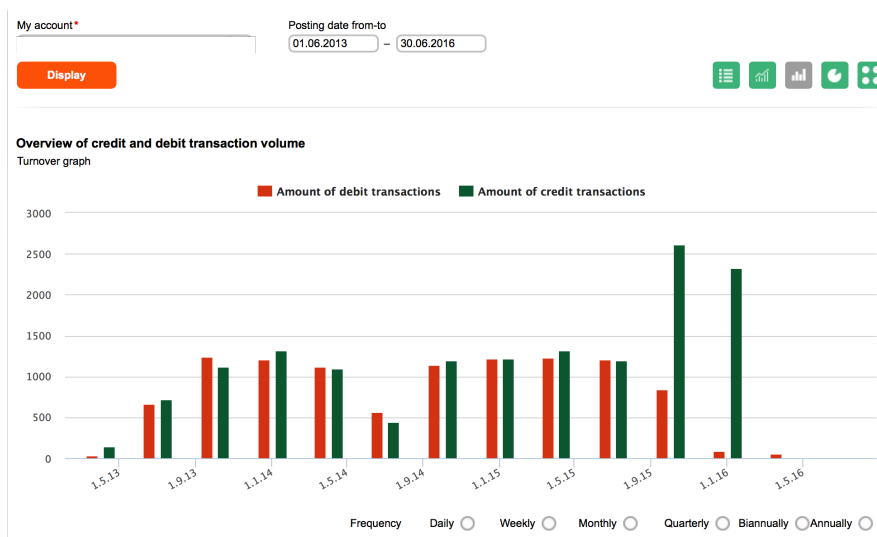
1. EXISTUJÚCE RIEŠENIA



Obr. 1.1: Webová aplikácia Fio banky

1.1.2 VÚB banka

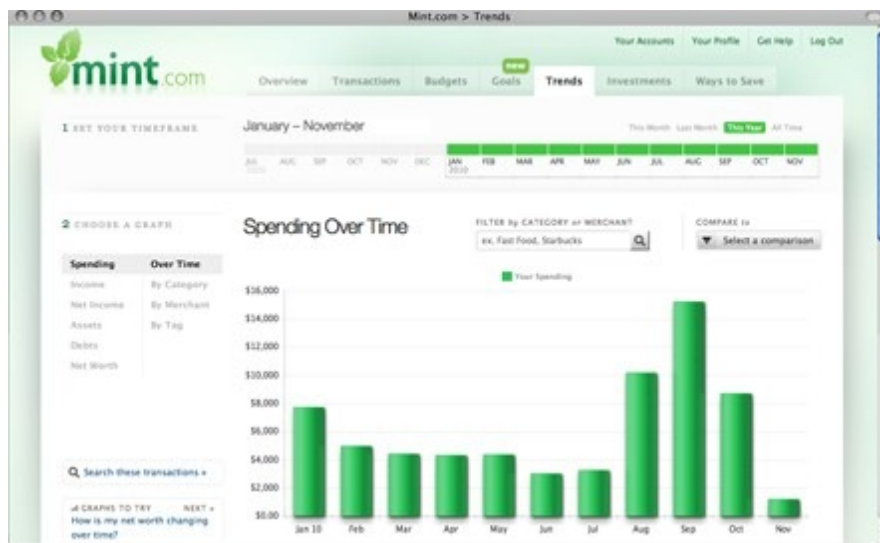
Vub.sk ponúka širšie možnosti reprezentácie dát. V ponúkaných grafoch nájdeme graf zostatku na účte, stĺpcový graf, ktorý zobrazuje príjmy a výdaje pri oboch grafoch je možnosť nastaviť časové obdobie a granularitu (Obrázok 1.2). Táto banka ponúka aj možnosť zaradiť jednotlivé transakcie do kategórií 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 aplikácií je hlavne uzavretosť čo je nevýhodou pre užívateľa hlavne z dôvodu analýzy jeho bankových transakcií poskytovateľom aplikácie. Ďalšou nevýhodou je nedostatočná podpora bánk na európskom trhu.



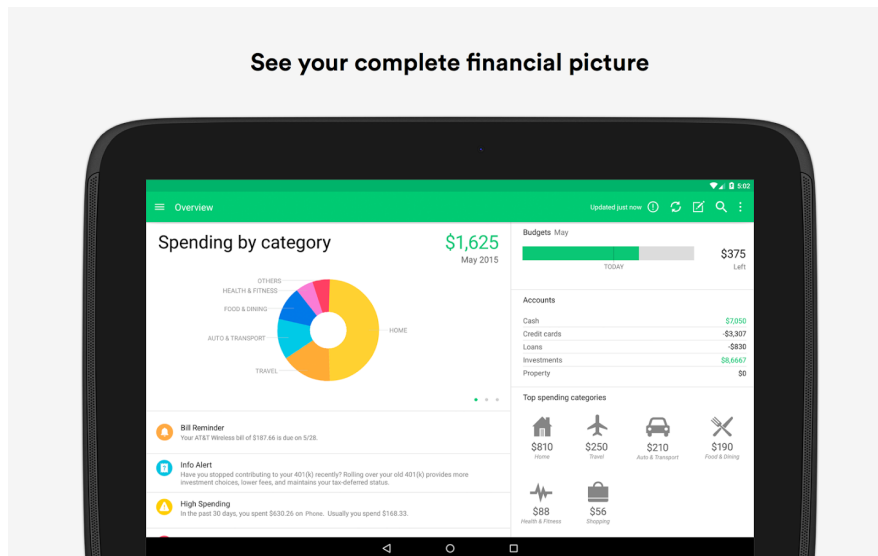
Obr. 1.3: Webová aplikácia mint.com

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

1.3 Mobilné aplikácie

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

1. EXISTUJÚCE RIEŠENIA



Obr. 1.4: Mobilná aplikácia mint.com

- + Jednoduchosť ovládania
- + Kategorizácia dát
- Pripojenie k internetu
- Niektoré banky nie sú podporované (chýba podpora CZ/SK bánk)
- Nie sú open-source

1.4 Šablóny

Je možné bankové výpisy spracovávať v štatistických programoch ako napríklad MS excel alebo v štatistickom jazyku ako napríklad R alebo wolfram alpha. Toto riešenie vyžaduje znalosť jedného z prostredí a je cielené skôr na skúsenejších užívateľov.

- + Možnosť pokročilejších štatistických metód
- + Open-source
- Nevhodné pre bežných užívateľov

1.5 Zhrnutie

Existuje veľa aplikácií na správu financií, väčšina má nieaký nedostatok či už ide o uzavrenosť aplikácie, nedostatok funkcionality alebo neschopnosť analyzovať dáta z viacerých zdrojov. Preto vzniká potreba vyvinúť aplikáciu, ktorá

bude stand-alone, open-source z možnosťou ju rozširovať o ďalšie bankové inštitúcie.

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 implementácii, 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 kapitola 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 spracovať, uložiť a znova načítať.

F3 - Program bude upozorňovať na duplikátne záznamy.

F4 - Program bude generovať grafy z nahratých dát.

Analýza požiadavku F1

Požiadavok F1 určuje, že môj program musí byť schopný importovať súbory v strojovo čitateľnom formáte. Asi najčastejšou formou výpisu z účtu s ktorou sa v bankovom sektore stretneme je formát CSV, no to nie je jediný formát ktorý

2. ANALÝZA

by mala aplikácia podporovať. V nasledujúcej 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], problém je že norma sa často nedodržiava kvôli zámene oddelovačov. S týmto formátom sa stretneme asi 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 dátové typy. Xml bol navrhnutý v roku 1996 a primutý ako W3C štandard v roku 1998 [2]. Tento formát nie je moc využívaný v bankovom sektore, ale stavajú na ňom iné formáty ako napríklad OFX ktorý je štandardom pre export bankových výpisov do štatistických programov.

- + Možnosť reprezentovať hierarchickú štruktúru 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 schopné reprezentovať hierarchickú štruktúru dát a dátové typy. JSON je štandardizovaný normou RFC 7159 [3]. Tento formát je využívaný pri stahovaní dát z API bankovej inštitúcie.

- + Možnosť reprezentovať hierarchickú štruktúru 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 transakcií.

2.1.2 Analýza požiadavku F2

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

2.1.3 Analýza požiadavku F3

Funkčný požiadavok 3. definuje problém duplikátnych záznamov ktoré sú nežiadúce pri zobrazovaní grafov, lebo by prezentovali užívateľovi skreslené dáta, 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 záznamy, ktoré sú vyhodnotené ako duplikáty a rozhodne o ich ponechaní alebo zmazaní.

2.1.4 Nefunkčné požiadavky

N1 - Ľahká rozšíriteľnosť

N2 - Stand-alone aplikácia bežiaca vo webovom prehliadači, podporované prehliadače: Google Chrome (verzia 30.0 a vyššie), Mozilla Firefox (verzia 24.0 a vyššie), Internet Explorer (verzia 10.0 a vyššie).

N3 - Intuitívne ovládanie.

N4 - Validácia vstupou.

Analýza požiadavku N1

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

Analýza požiadavku N2

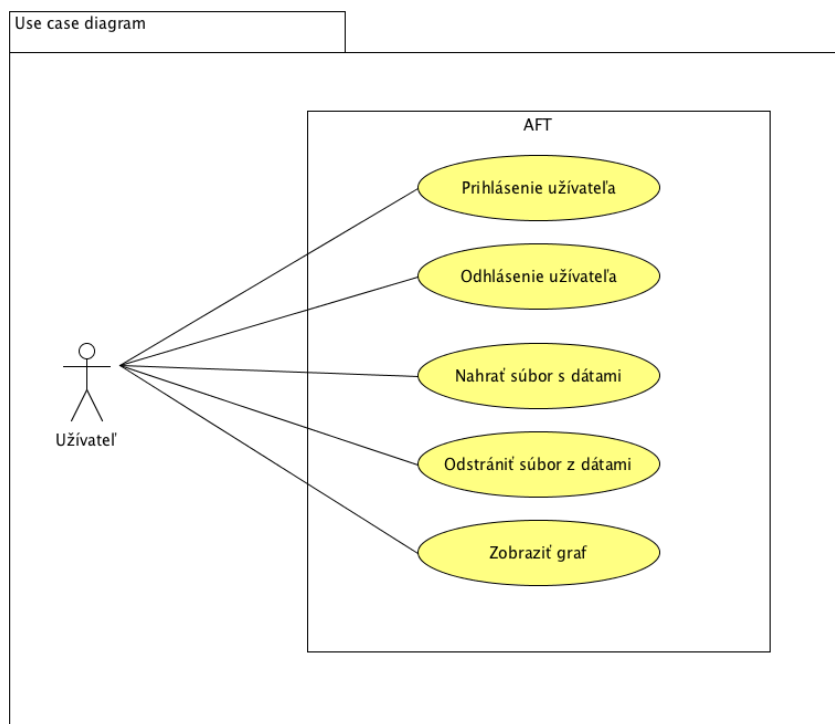
Aplikácia bude bežať vo webovom prehliadači z dôvodu multiplatformosti, bezpečnosti a jednoduchosti užívateľského rozhrania. Bude sa jednať o stand-alone aplikáciu z dôvodu citlivosti spracovávaných dát ktoré nechceme odovzdávať 3 stranám. Aplikácia nebude potrebovať k nasadeniu webový server kvôli jednoduchosti nasadenia aplikácie.

Analýza požiadavku N3

Aplikácia bude mať intuitívne ovládanie, 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 potom detailne rozobraté do prípadov použitia. Tiež popisuje rôznych používateľov systému.



Obr. 2.1: Diagram prípadou použitia programu

2.2.1 Popisy prípadov použitia

1. Prihlásenie užívateľa

- Užívateľ zadá meno a heslo do formulára, aplikácia overí správnosť a nahrá jeho uložená dáta.

2. Odhlásenie užívateľa

- Aplikácia zmaže všetky užívateľské dáta z dočasnej pameti.

3. Nahráť súbor z dátami

- Užívateľ zvolí súbor ktorý sa má nahráť do aplikácie, a zadá meno banky do formulára.

4. Odstránenie súboru z 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.

2.2.2 Zoznam účastníkov

Na zoznamu účastníkov sú dvaja užívatelia, ktorí sa od seba odlišujú len prihlásením. Užívateľ importuje nové dáta, pokiaľ je prihlásený, 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 a súbor zo systému.
4. Aplikácia následne spracuje súbor a vyhodnotí či sa v nom nenachádzajú duplikátne záznamy s niektorým z už nahratým súborom, ak áno aplikácia navrhne užívateľovi odstánenie 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.

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 implementovať 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, Mena. Ale môžu obsahovať aj zaujímavejšie dáta ako napríklad poloha alebo kategória transakcie, tieto extra dáta sú veľmi zaujímavé ale neposkytuje ich každá banka a niesú štandardizované, takže 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. Problém s týmito informáciami je v tom že sa nachádzajú v neštandarizovanej poznámke, ktorú je náročné spracovať. Po zvolení časového úseku zo zoznamu a položiek 2.2 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, takže nemôžeme očakávať jednotnú štruktúru dát.
- CSV API
 - Táto možnosť úplne ignoruje užívateľovu voľbu položiek a dodržiava 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.

<input checked="" type="checkbox"/> Akce	<input type="checkbox"/> Protiúčtet
<input checked="" type="checkbox"/> Částka	<input type="checkbox"/> Reference plátce
<input checked="" type="checkbox"/> Datum	<input type="checkbox"/> SS
<input type="checkbox"/> ID operace	<input type="checkbox"/> Symboly
<input type="checkbox"/> ID pokynu	<input checked="" type="checkbox"/> Typ
<input type="checkbox"/> KS	<input checked="" type="checkbox"/> Upřesnění
<input checked="" type="checkbox"/> Název banky	<input type="checkbox"/> VS
<input type="checkbox"/> Název protiúčtu	<input type="checkbox"/> Zadal
<input checked="" type="checkbox"/> Poznámka	<input type="checkbox"/> Zpráva pro příjemce

Obr. 2.2: Formulár exportovaných položiek Fio.cz

2. ANALÝZA

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]. Problémom je zložitá deserializácia.
- CSV
 - Zvolený formát kvôli jednoduchosti importu dát.
- XML
- XLS
 - Tabuľka vo formáte vhodnom na spracovanie v Microsoft Excel.

The screenshot shows the 'My account*' section with a blacked-out account number. It includes filters for 'Partner's account number', 'Execution date from-to' (11.05.2015 to 08.05.2019), and 'Amount from-to'. Below these are 'Other options to filter data' including 'Transaction type*' (All), 'BIC', 'Partner's reference', 'Payment partner', 'Additional information', 'Posting date from-to', 'Variable symbol', 'Specific symbol', 'Constant symbol', 'Document number', 'Identifier of a payment batch', and 'Payment category' (Living). There are 'Search' and 'Cancel filter' buttons. At the bottom is a table of transactions.

Posting	Effective date	Amount	Type	Partner's account	Partner's name	Partner's reference	Additional information	Balance
27.12.2016	22.12.2016	-545.88 €	📄		VUB,EVI...		DDNAY ELEKTRODO...	3,559.94 €
11.03.2016	09.03.2016	-86.03 €	📄		VUB,EVI...		DDALZA SHOWROO...	4,156.67 €

Obr. 2.3: Formulár exportu Vub.sk

2.4 Analýza typov grafov

2.4.1 Stĺpcový graf



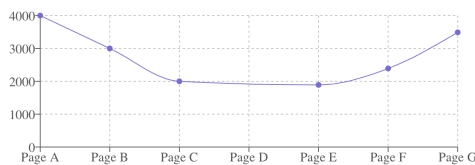
Obr. 2.4: Stĺpcový graf

Stĺpcový graf zobrazuje dáta za použitia horizontálnych alebo vertikálnych stĺpcov, ktoré porovnávajú diskkrétne veličiny. Jedna os zobrazuje jednotlivé porovnávané kategórie, druhá hodnotu diskkrétnej veličiny. Tento typ grafu je používaný hlavne na zobrazenie kategorických dát. Kategorické dáta sú dáta rozdelené do diskkrétnych 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 transakcií ktoré boli vykonané počas definovaného časového úseku. Užívateľovi tento graf má načrtnúť trend výdajov na účte.
- Výdaje počas jednotlivých kategórií (Typy transakcií, Banky, Dni v týždni, Mesiace v roku).
 - Súčet všetkých transakcií ktoré spadajú do danej kategórie. Užívateľ bude schopný vyčítať z grafu ktoré položky sú najnákladovejšie.
- Transakcie v obchode.
 - Tento graf bude zobrazovať všetky výdaje vytvorené v jednom obchode. Pre užívateľa bude tento graf užitočný hlavne kvôli optimalizácii výdajov v niektorých z obchodov.

2.4.2 Čiarový graf

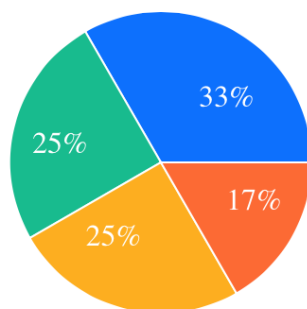


Obr. 2.5: Čiarový graf

Tento graf zobrazuje informáciu ako sériu bodov spojených čiarou, body sú najčastejšie v časovom poradí. Využíva sa na zobrazenie trendu v dátach počas nieakého časového obdobia. Tento graf nebudem používať na zobrazovanie zostatku na účte z dôvodu neexistencie záznamu o štartovnej bilancii v bankových dátach.

- 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 príjmou do daného dátumu.

2.4.3 Koláčový graf



Obr. 2.6: Koláčový graf

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

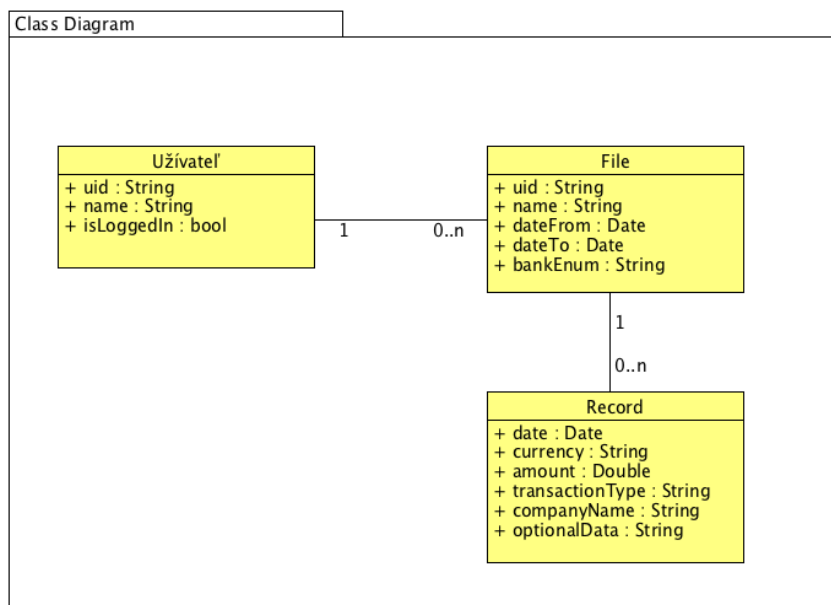
- Typy transakcií
- Banky

Návrh

Návrh je dôležitou fázou pri tvorbe softwaru. Vychádza z analýzy funkčných a nefunkčných požiadaviek. Táto etapa pozostáva z konceptuálneho modelu, databázového modelu a návrhu používateľského rozhrania.

3.1 Konceptuálny model

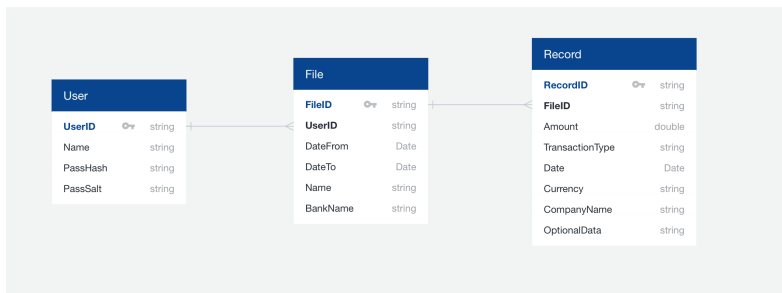
Konceptuálny model je zobrazený na obrázku 3.1. Hlavným prvkom aplikácie je užívateľ, ktorý môže mať niekoľko súborov ktoré obsahujú v sebe záznamy o transakciách.



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 databázy.



Obr. 3.2: Logický model databázy

3.3 Návrh používateľského rozhrania

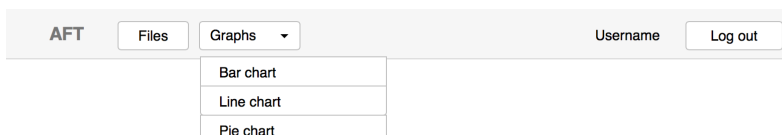
Pri návrhu webovej aplikácie je dôležité dbať na prehľadnosť a jednoduché používanie. Ešte pred samotnou implementáciou aplikácie je dobré zamyslieť sa nad rozložením komponent užívateľského prostredia. Na toto mi poslúžili takzvané wireframe diagramy, čiže diagramy ktoré pomôžu z rozvrhnutím ovládacích prvkov v aplikácii.

Aplikácia sa bude skladať z 2 hlavných častí:

- Hlavička
- Obsah

3.3.1 Hlavička

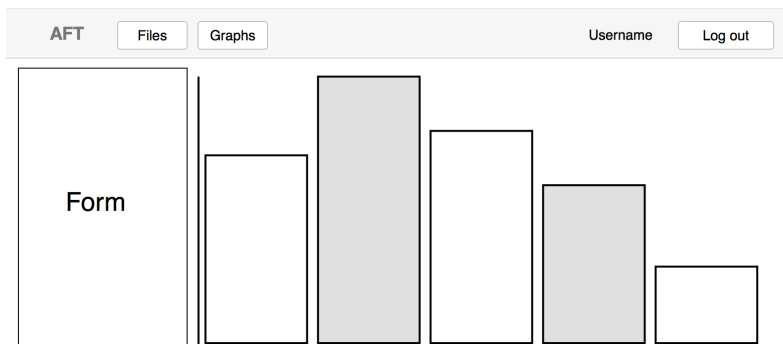
Hlavička bude obsahovať názov aplikácie, ktorá bude odkazovať na domovskú stránku. Navigácia stránky bude prebiehať pomocou odkazov Files a Graphs. Graphs bude obsahovať vysúvacie menu z ponukou grafov. Na pravej strane budú buď tlačítka na prihlásenie užívateľa a jeho registráciu alebo meno prihláseného a možnosť odhlásiť sa.



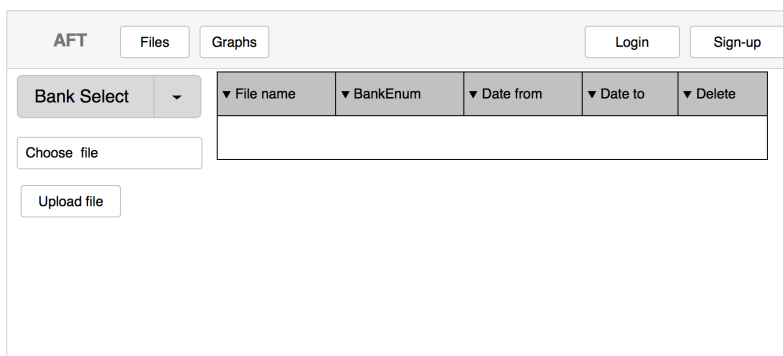
Obr. 3.3: Wireframe hlavičky

3.3.2 Obsah

Obsah sa bude meniť podľa aktuálne zvoleného náhľadu užívateľom. Buď sa bude jednať o zobrazenie nahraných súborov spolu s formulárom v ktorom je možnosť zvoliť banku a súbor z počítača obr 3.4, alebo zobrazenie vybraného grafu z formulárom na nastavenie filtra obr 3.5.



Obr. 3.4: Wireframe zobrazenia grafu



Obr. 3.5: Wireframe zobrazenia súborov

Realizácia

V tejto kapitole priblížim realizáciu navrhovanej aplikácie a nakoniec zhrniem do akého štádia sa mi podarilo aplikáciu implementovať.

4.1 Vybrané technológie

Podľa nefunkčných požiadaviek z analýzy, navrhovaná aplikácia má byť webová, pre používateľa jednoducho použiteľná a responzívna. Program bude bežať vo webovom prehliadači, takže budem samozrejme používať jazyk HTML spolu s CSS na zobrazovanie užívateľského prostredia. Aplikačnú logiku bude zabezpečovať programovací jazyk JavaScript, na ktorom sú postavené knižnice, ktoré bude program využívať.

4.2 Zvolené knižnice

4.2.1 React

React je open-source knižnica vyvíjaná Facebookom. Slúži na tvorbu užívateľského rozhrania. Jednou z hlavných výhod je schopnosť updatovať obsah aplikácie bez nutnosti znova načítať stránku. Jedná sa o relatívne novú knižnicu, ale z dôvodu jednoduchosti vývoja a podporou veľkých spoločností ako je napríklad Twitter alebo Facebook si veľa vývojárov volí React ako základ svojich webových aplikácií.[7]

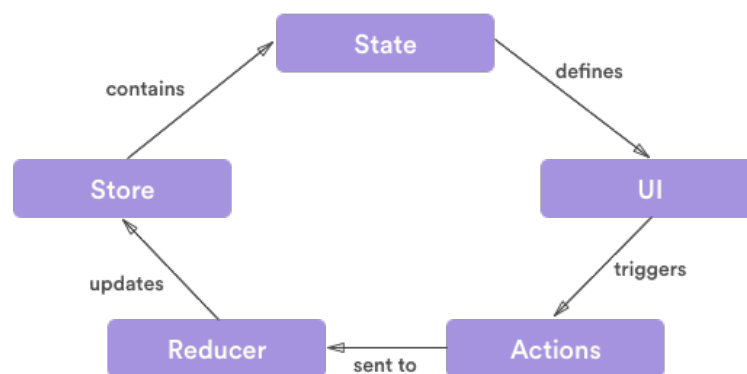
4.2.2 Redux

Redux ponúka možnosť udržiavať aktuálny stav aplikácie v jednom kontajneri, z ktorého sa generuje užívateľské prostredie. Knižnica zjednodušuje vývoj jednoduchým testovaním, ponúka možnosť testovania formou cestovania po jednotlivých stavoch aplikácie.[8]

4. REALIZÁCIA

Skladá sa z 3 častí:

- Store - Kontajner ktorý obsahuje stav aplikácie.
- Actions - Akcie ktoré vyvoláva užívateľ a vysielajú dáta reducerom.
- Reducers - Upravujú obsah store podľa dát prijatých z akcií.



Obr. 4.1: Štruktúra redux aplikácie

4.2.3 React-Bootstrap

Túto knižnicu som sa rozhodol použiť kvôli jednoduchosti implementácie a ešte aj preto lebo ponúka responzivnosť v nej nadefinovaných prvkov. Jedná sa o port prednej front-endovej knižnice do reactu. Obsahuje veľa štýlov pre bežné HTML prvky ako napríklad tlačítka, tabuľky, formuláre atď.. Ale implementuje aj vlastné komponenty ako napríklad navigačné listy. [9]

4.2.4 IndexedDB

IndexedDB je nízko úrovňovým API ktoré je implementované vo väčšine moderných prehliadačov. [10]

4.2.5 Dexie.js

Knižnica ktorá stavia databázu nad IndexedDB a ponúka efektívny spôsob spravovania a vyhľadávania v nej. Táto knižnica ďalej ponúka možnosť spojenia z dátábázovým servrom kde sa dá zálohovať lokálna verzia databázy, táto funkcionality nebude implementovaná v aplikácii z dôvodu potreby inštalácie webového serveru. [11]


4.2.6 Iné

- reCharts.js - Knižnica na generovanie grafov [12]
- moment.js - Parsovanie a formátovanie dátumov [13]
- papaparse.js - Parsovanie CSV súborov [14]
- money.js - Knižnica zabezpečujúca konverziu mien v rámci aplikácie [15]
- crypto.js - Implementuje rôzne hashovacie a šifrovacie štandardy do javascriptu

4.3 Tvorba používateľského rozhrania

Tvorba používateľského rozhrania bola celkom jednoduchá, react-bootstrap ponúka veľké spektrum komponentov a má dobre spracovanú dokumentáciu. Podľa nefunkčných požiadaviek má byť používateľské rozhranie responzívne čo zabezpečuje komponenta grid ktorá delí aplikáciu do 12 stĺpcov a dovoľuje nadefinovať rozmery jednotlivých prvkov v návrhu pre jednotlivé zariadenia. Responzivnosť demonštrujem na obrázku 4.2.

The image displays two parts of a web application. The top part is a dark sidebar menu for 'Bank statement analyzer' with a hamburger menu icon. The bottom part is a light-colored login form with fields for Name, Password, and Repeat password, and a Sign in button.

Bank statement analyzer 

Files

Graphs ▼


Login

Sign up

Name

Password

Repeat password

Sign in

Obr. 4.2: Ukážka responzívnosti aplikácie

4.4 Správa užívateľských dát

Užívateľské dáta udržiavam v databáze IndexedDB. Vzhľadom na to že aplikácia podporuje väčšie množstvo užívateľov, je nutné spravovať užívateľské mená a heslá. Tu vzniká otázka bezpečnosti ukladania hesiel. Užívateľské mená ukladám v plain texte ale heslá hashujem pomocou algoritmu PBKDF2 ten bere ako vstupné parametre heslo, soľ a počet iterácií.

4.5 Validácia dát

Prvá validácia prebieha už pri výbere súboru, musí sa jednať o textový súbor z jednou zo známych prípon. Do validácie dát spadá aj filtrovanie duplikátnych záznamov z užívateľových transakcií. To riešim hashovaním povinných parametrov transakcie pomocou SHA512 a porovnávam ich so záznamami nachádzajúcimi sa v databáze. Pokiaľ sú nájdené duplikáty, upozorním na nich užívateľa a dám mu na výber ako ich spracovať. Samotnú validáciu dát v súbore nechávam na modul banky, ktorá má zadaný súbor spracovať.

4.6 Konverzia kurzov

Na konverziu kurzov používam knižnicu money.js, kurzy sú stiahnuté z Open Exchange Rates pokiaľ inicializácia zlyhá tak knižnica použije kurzy uložené v databáze. V programe sa všetky mená mien riadia štandardom ISO 4217.

4.7 Vykreslenie grafu

Grafy sú implementované ako samostatné objekty ktoré spravujú formulár ako aj samotné vykresľovanie. Po tom čo si užívateľ zvolí typ grafu a vyplní dáta vo formulári tak sa zavolá filter príslušného grafu, updatujú sa dáta v store a vykreslí sa graf. Pri implementácii bolo prihliadnuté na jednoduchosť pridania ďalších grafov.

```
1 Graph.prototype = {
2   //name of graph displayed in graph offer
3   name: String,
4   //unique graph enum
5   enum: String,
6   //returns graphHandler
7   getHandler() {
8     return new GraphHandler();
9   }
10 }
```

Listing 4.1: Graph

```
1 GraphHandler.prototype = {
2   //returns filter generated from form
3   getFilter(),
4   //renders form, calls UpdateGraphDataFunction when update of
      display data needed
5   renderForm(updateGraphDataFunction),
6   //filters transactions according to filter and returns data
      to be displayed by renderGraph()
7   filterRecords(records, filter),
8   //renders graph based on data supplied in graphData
9   renderGraph(graphData)
10 }
```

Listing 4.2: GraphHandler

4.8 Implementácia bankových modulov

V mojej bakalárskej práci som sa zaoberal implementáciou modulou pre fio.cz a vub.sk, pri implementácii parsovania dát som sa riadil špecifikáciami API [5] [6]. Bankový modul má v sebe nadefinovanú funkciu, ktorá parsuje vstupný súbor danej banky. Ďalšie moduly sú tak len jednoduché objekty s parsovacou funkciou.

```
1 Bank.prototype = {
2   //next id in line
3   id: Number,
4   //name to be displayed in selection
5   name: String,
6   //enum of the bank institution (web page in caps with out
      symbols)
7   enum: String,
8   //color that will be displayed in graph
9   color: String,
10  //parsing method receives contents of file
11  parse(data);
12  //custom graphs for bank
13  graphs: Array of Graph
14 }
```

Listing 4.3: Bank

4.9 Zhrnutie

Pri implementácii som sa držal návrhu a wireframov, ktoré boli veľkou pomocou pri tvorení užívateľského prostredia.

Najväčší problém počas implementácie som mal z nájdením databáze, ktorá by bola perzistentná. Vzhľadom na to, že webové prehliadače fungujú ako sandbox tak nemajú prístup k suborovému systému. Nakoniec som na-

razil na IndexedDB, ktorá do istej miery splňuje všetky požiadavky. Jedinou chybou tejto databáze je že nepodporuje šifrovanie.

Testovanie

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. Túto možnosť som bohužiaľ nemal, takže som kód po sebe kontroloval sám.

5.2 Statická analýza kódu

Pri statickej analýze kódu sa hľadajú chyby v návrhových vzoroch, ako napríklad zachytávanie výnimok alebo nedosažiteľnosť niektorých z vetiev. Vzhľadom na to že som pri vývoji používal WebStorm, ktorý túto analýzu vykonáva automaticky, nemusel som statickú analýzu kódu vykonávať ručne.

5.3 Testovanie nefunkčných požiadaviek

V týchto testoch som hlavne overoval funkčnosť aplikácie na všetkých webových prehliadačoch nadefinovaných v nefunkčných požiadavkách.

5.4 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.

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 taktiež zvolil implementačnú platformu, navrhol databázu a vytvoril wireframy. Následne som v realizácii popísal ako prebiehala implementácia a riešenia, ktoré som zakomponoval do aplikácie.

Výsledkom tejto bakalárskej práce je stand-alone aplikácia bežiaca vo webovom prehliadači, ktorá spracováva bankové výpisi. Týmto programom bol splnený cieľ práce.

Ale stále sú tu možnosti rozšírenia aplikácie o ďalšie grafy a banky. Jedno z možných rozšírení, ktoré som nestihol zakomponovať do výslednej aplikácie je pridanie automatickej kategorizácie transakcií. Ďalším zaujímavým spôsobom rozšírenia by bola implementácia databázového servera, ktorý by zálohoval dáta. Táto funkcionality by bola zaujímavá pre skúsenejších užívateľov.

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 Bankovníctví. [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] React. [online], [cit. 2017-05-12]. Dostupné z: <https://facebook.github.io/react/>
- [8] Redux. [online], [cit. 2017-05-12]. Dostupné z: <http://redux.js.org>
- [9] React-bootstrap. [online], [cit. 2017-05-12]. Dostupné z: <https://react-bootstrap.github.io>
- [10] IndexedDB. [online], [cit. 2017-05-12]. Dostupné z: <https://www.w3.org/TR/IndexedDB/>
- [11] Dexie.js. [online], [cit. 2017-05-12]. Dostupné z: <http://dexie.org/>

LITERATÚRA

- [12] Recharts. [online], [cit. 2017-05-12]. Dostupné z: <http://recharts.org/>
- [13] moment.js. [online], [cit. 2017-05-12]. Dostupné z: <https://momentjs.com>
- [14] Papaparse. [online], [cit. 2017-05-12]. Dostupné z: <http://papaparse.com>
- [15] money.js. [online], [cit. 2017-05-12]. Dostupné z: <http://openexchangerates.github.io/money.js/>

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

Obsah priloženého CD

	readme.txt.....	stručný popis obsahu CD
	exe.....	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