VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

APLIKACE PRO ANALÝZU DAT Z BANKOVNÍCH TRANSAKCÍ

APPLICATION FOR ANALYSIS OF DATA FROM BANK TRANSACTIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PETR HÝBL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

147943

Ústav:	Ústav informačních systémů (UIFS)
Student:	**Hýbl Petr**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Aplikace pro analýzu dat z bankovních transakcí**
Kategorie:	Informační systémy
Akademický rok:	2022/23

Zadání:

1. Seznamte se s problematikou tvorby webových aplikací, dostupnými prostředími a frameworky. Dále se seznamte s formáty dat, ve kterých poskytují jednotlivé banky své výpisy.
2. Analyzujte požadavky na aplikaci pro import dat z bankovních výpisů, která bude dále umožňovat jejich podrobnou analýzu včetně možnosti automatického určení typů jednotlivých položek. Požadavky konzultujte s vedoucím.
3. Navrhněte aplikaci splňující požadavky, použijte k tomu vhodné prostředky jazyka UML.
4. Implementujte navrženou aplikaci a proveďte testování na vhodném vzorku dat zahrnující různé formáty vstupních dat.
5. Zhodnoťte dosažené výsledky a další možnosti pokračování v tomto projektu.

Literatura:

- Vollset, E., Folkestad, E., Gallala, M.R., Gulla, J.A.: Making Use of External Company Data to Improve the Classification of Bank Transactions. In: Advanced Data Mining and Applications. ADMA 2017. LNCS, vol. 10604. Springer.
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9.
- Welling, L., Thomson, L.: PHP a MySQL: Kompletní průvodce vývojáře. CPress, 2017.

Při obhajobě semestrální části projektu je požadováno:
Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	24.10.2022

Abstrakt

Cílem této práce je navrhnout a implementovat webovou aplikaci pro analýzu bankovních transakcí. Aplikace využívá na serveru webový framework Flask a na klientu framework React. Při práci byl kladen důraz zejména na jednoduchost a uživatelský zážitek. Hlavní funkcí této aplikace je automatická kategorizace transakcí pomocí umělé inteligence. Podstatou této funkce je dvouvrstvá architektura. V této architektuře se podařilo dosáhnout úspěšnosti 85 %. Tato aplikace umožňuje uživatelovi nahrát měsíční výpis a zobrazit si přehledný graf s kategorií, kde nejvíc utrácí. Dále může uživatel porovnávat jednotlivé měsíce a přemýšlet nad tím jak optimalizovat své finance.

Abstract

The goal of this thesis is to design and implement a web application for analyzing bank transactions. The application uses the Flask web framework on the server and the React framework on the client. The focus of the work was simplicity and user experience. The main function of this application is automatic categorization of transactions using artificial intelligence. The core of this feature is a two-layer architecture. In this architecture we were able to achieve a success rate of 85 %. This application allows the user to upload a monthly statement and view a summary graph with the category where they spend the most money. In addition, the user can compare each month and think about how to optimize their finances.

Klíčová slova

webová aplikace, bankovní transakce, analýza umělou inteligencí, React, Javascript, Python, Flask, bankovní výpisy

Keywords

web application, bank transaction, analysis by artificial intelligence, React, Javascript, Python, Flask, bank statements

Citace

HÝBL, Petr. *Aplikace pro analýzu dat z bankovních transakcí.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Aplikace pro analýzu dat z bankovních transakcí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Petr Hýbl
7. května 2023

Poděkování

Děkuji vedoucímu práce panu Ing. Vladimírovi Bartíkovi, Ph.D. za veškeré odborné konzultace a poskytnuté rady v celém průběhu tvorby této práce. Poděkování taktéž patří mé rodině a kamarádům, kteří mi poskytli jejich bankovní výpisy.

Obsah

Kapitola 1

Úvod

Zlepšení finanční gramotnosti v České republice je stále téma, které mnoho odborníků neví jak řešit. Moje motivace není v celkovém řešení společenského problému, nýbrž chci přivést na trh nástroj, který dokáže pomoci běžnému občanu České republiky v každodenním rozhodování o jeho financích. Sestavením rodinného rozpočtu se zabývám několik posledních let a tato zkušenost mě přivedla k vytvoření této práce a pomoci nejen sobě, ale i ostatním v oblasti finanční zodpovědnosti. Zároveň zde mohu zúročit, vše co mě naučila vysoká škola v oblasti umělé inteligence a programování webových aplikací.

Proto jsem se rozhodl vytvořit tuto práci a vytvořit nekomerční řešení pro správu rodinných financí.

Cílem této bakalářské práce je vytvořit webovou aplikaci, která bude analyzovat transakce a řadit je do předem určených kategorií. Po registraci uživatelovi tato aplikace dovolí nahrát své měsíční bankovní výpisy. Uživatel si vybere svou bankovní společnost, díky které se aplikuje šablona formátu výpisu na jeho formát.

Následně se extrahují jednotlivé transakce a pomocí umělé inteligence se roztřídí do různých obecných kategorií, např. jídlo, transport, potraviny a energie. Následně se tyto transakce zobrazí v přehledném grafu s informacemi, ve které sekci má nejvyšší výdaje. Součástí aplikace bude také ruční úprava transakcí, pokud umělá inteligence špatně zvolí kategorii nebo nedokáže sama tuto kategorii zařadit. Uživatel může i ručně odebírat transakce a prohlížet si grafy z minulých měsíců. Aplikace si bude zakládat na uživatelské přívětivosti a jednoduchosti používání.

V první kapitole shrnu a rozeberu současné aplikace na trhu a důvody, proč vytvářím aplikaci novou. Každou aplikaci shrnu v pár větách a zdůvodním její nedostatky.

Prvním bodem mého zadání bylo se seznámit s problematikou vytváření webových aplikací. Součástí je i následné vyhodnocení a výběr vhodného frameworku a architektury, která bude použita pro realizaci výsledné aplikace.

V další kapitole se seznámíme s dostupnými formáty výpisů, které poskytují přední bankovní instituce v České republice a provedu analýzu požadavků pro naši aplikaci pomocí diagramu případů užití.

Druhým bodem je analýza popisu transakcí a extrakce údajů. Tato data poté využijeme na kategorizaci transakcí umělou inteligencí. Následujícím krokem je sestavení vhodného modelu pro umělou inteligenci a jeho následné trénování na testovacích datech.

Třetí část souvisí s návrhem výsledné aplikace a s tímto bodem také spojená tvorba ER diagramu. Detailně zde bude popsán návrh uživatelského rozhraní a API.

V následující části uvedu, jak vypadá implementace samotné aplikace a rozeberu automatickou klasifikaci transakcí. V předposlední kapitole shrnu výsledky testování celé aplikace a vyzkouším aplikaci testováním na několika lidech.

V poslední kapitole shrnu výsledky práce, zmiňuji se i o její použitelnosti a také o možnostech pro její budoucí rozvoj.

Kapitola 2

Analýza průzkumu trhu

V následující kapitole budou popsány nejznámější aplikace pro správu financí.

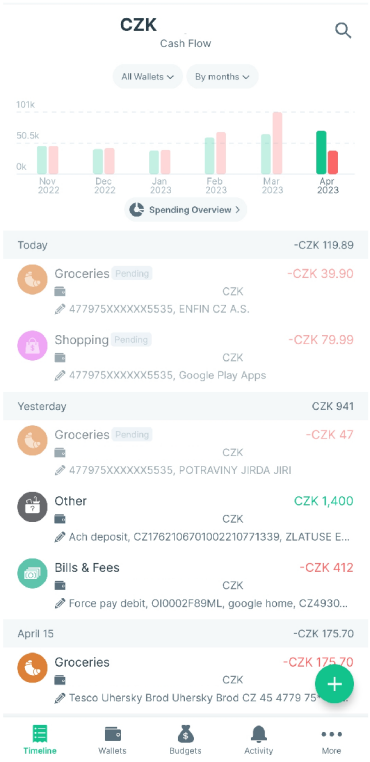
2.1 Mobilní aplikace

V dnešní době jsou mobilní aplikace důležitější a rozšířenější, než kdy dříve. V oblasti financí jsou mobilní aplikace stále důležitější pro uživatele, kteří chtějí být informováni o stavech na bankovních účtech a plánovat své výdaje. Trh s mobilními aplikacemi, které se zabývají rozpočtem je opravdu obrovský, každá umí něco jiného a má svá specifika, jak ji používat.

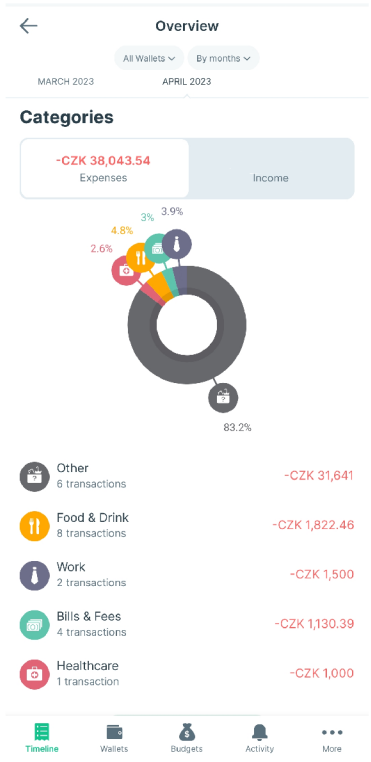
Spendee

Nejoblíbenější aplikací na trhu je aplikace Spendee[1]. Aplikace dokáže sledovat vaši útratu pomocí propojení na bankovní API. Jako jediná aplikace se dokáže napojit i na české banky. Tato služba zní velmi zajímavě bohužel její hlavní nevýhoda je, že je placená. Měsíční předplatné stojí buď 2$ nebo dražší 3$. Aplikace je na trhu již od roku 2018. Spendee nabízí uživatelům také přehledné statistiky o jejich výdajích a příjmech. Aplikace zobrazuje grafy a tabulky, které umožňují uživatelům vidět, jak se stav jejich financí vyvíjí v průběhu času. Díky tomu mohou uživatelé lépe plánovat své budoucí výdaje a sledovat, jak se jim daří hospodařit s penězi. Aplikace cílí na masovou společnost, která chce jednoduché ovládání a pěkné barevné GUI. Během testování jsem zaznamenal problémy s propojením s Komerční bankou, kdy aplikace často přerušovala spojení. Spendee je určitě skvělou volbou pokud chceme sledovat naše finance a nevadí nám časté přihlašování do bankovního API. Rozhraní aplikace můžeme vidět na obrázku 2.1 a rozvržení grafu na obrázku 2.2.

[1] https://www.spendee.com/



Obrázek 2.1: Aplikace Spendee souhrn výdajů.



Obrázek 2.2: Aplikace Spendee souhrn graf.

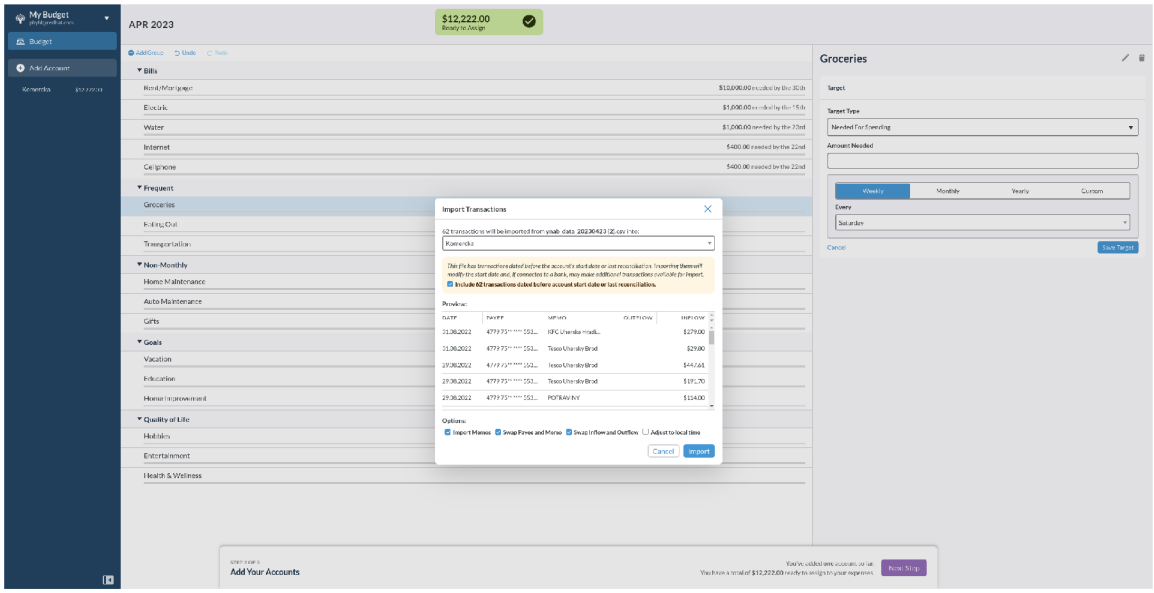
YNAB (You need a budget)

YNAB[2] (You Need A Budget) je aplikace pro správu osobních financí, která pomáhá uživatelům získat kontrolu nad svými financemi a plánovat své výdaje. Tato aplikace se zaměřuje na tzv. pravidlo 4 kroků, které zní:

- Přiřazuj každý dolar k nějakému výdaji.
- Sleduj všechny výdaje.
- Stanov si priority pro výdaje.
- Zůstávej v mezích svého rozpočtu.

YNAB pomáhá uživatelům žít podle svého rozpočtu a plánovat si své výdaje s předstihem, což umožňuje snížit stres z financí a dosáhnout finanční svobody. Aplikace stojí 15$ měsíčně a nemá žádnou neplacenou verzi. Do aplikace si postupně zapíšete svoje měsíční výdaje a následně vás aplikace požádá o doplnění transakcí. Aplikace také nabídne napojení na API banky, ale v seznamu se nenachází žádná česká banka. Aplikace umožní nahrání CSV souboru pouze přes webové prostředí, které je značně nepřehledné a je nutné upravit CSV soubor, který nám poskytne banka. Na obrázku 2.3 můžeme vidět modální okno pro nahrání souboru.

[2] https://www.ynab.com/



Obrázek 2.3: Nahrávání výpisu do YNAB.

2.2 Webové aplikace

Webové aplikace jsou programy, které běží v internetovém prohlížeči uživatele a umožňují interakci s obsahem na internetu. Tyto aplikace jsou stále populárnějšími, neboť jsou snadno dostupné, nemusí se instalovat a mohou být používány na jakémkoli přístroji s připojením k internetu. Ve světě financí si lidé zvykli na internetové bankovnictví. Byla to první možnost komunikace s bankou bez bankéře a lidé je používají i dnes, a právě proto si myslím, že v dnešní době dává smysl mít finanční aplikace dostupné na webu.

2.2.1 Mint

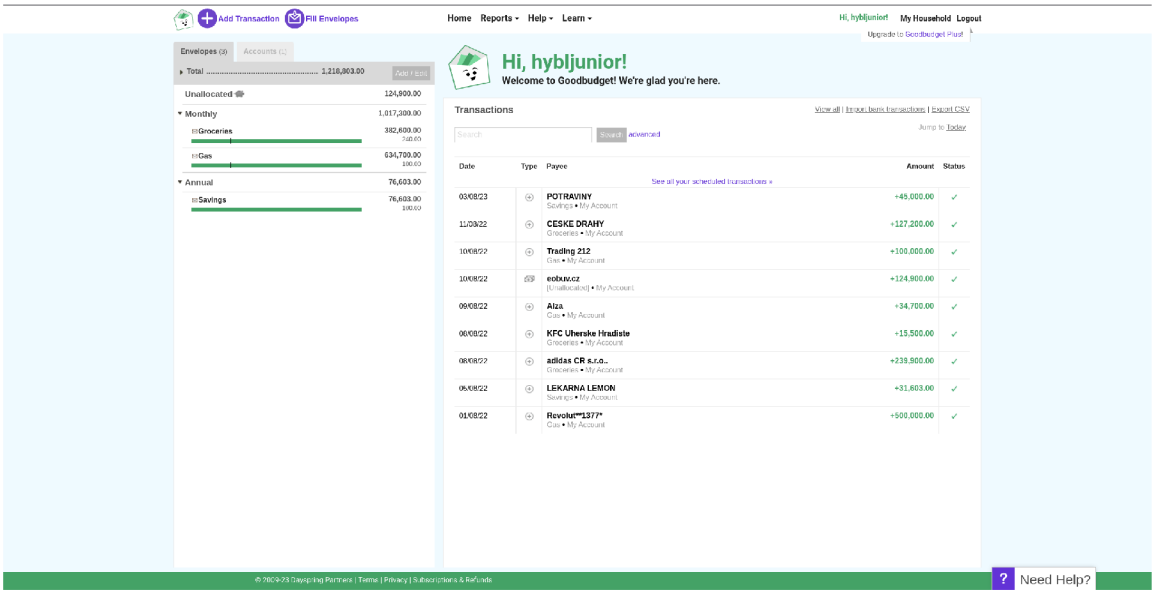
Aplikace Mint[3] je nástroj pro správu financí, která slouží ke sledování a správě vašich osobních financí. Tato aplikace vám umožňuje propojit vaše bankovní účty, kreditní karty, investiční účty a další finanční účty, aby vám poskytla celkový přehled o vašich finančních aktivitách. Mint vám umožňuje sledovat vaše příjmy a výdaje, kategorizovat transakce a sledovat vaše výdaje v reálném čase. Aplikace vám také poskytuje různé nástroje a funkce, např. upozornění na zůstatek na účtu, sledování kreditního skóre, plánování rozpočtu a investiční nástroje. Webová aplikace není dostupná v České republice, nepodporuje banky působící na tuzemském trhu, ale v zahraničí je velmi oblíbená, podobně jako Spendee. Také zde přidáváme své transakce po jedné a následně vidíte grafy, kde se co dá zlepšit. Aplikace působí přehledně.

2.2.2 Goodbudget

Goodbudget[4] je aplikace pro správu osobních financí, která pomáhá uživatelům plánovat své výdaje pomocí obálkové metody. Tato metoda spočívá v tom, že uživatelé si přiřadí k jednotlivým kategoriím své výdaje do obálky s určitými peněžními limity a snaží se

[3] https://mint.intuit.com/
[4] https://goodbudget.com/

tyto limity nepřekročit. Goodbudget umožňuje uživatelům sledovat, kolik peněz jim zbývá v každé kategorii, a tím koordinovat následující výdaje. Aplikace také umožňuje sdílet účty s rodinou. Webová aplikace nabízí jednoduché nahrání CSV souboru, dokonce ve webovém prostředí nabídne úpravu CSV souboru. Moje testovací výpisy z Komerční banky, ale byly velmi špatně rozpoznány a ani v jedné transakci nebyla správná hodnota transakce, jak můžeme vidět na obrázku 2.4



Obrázek 2.4: Přehled transakcí v Goodbudget.

2.3 Závěr

Shrnul jsem zde 4 aplikace pro správu osobních financí, které jsou v dnešní době dle různých internetových magazínů nejpopulárnější a také jsem si je mohl osobně vyzkoušet. Největší negativa těchto aplikací jsou uživatelská nekomfortnost při nahrávání vašich výpisů a placení měsíčních poplatků. Z této analýzy jsem dospěl k závěru, že v České republice chybí nekomerční aplikace pro kategorizaci vašich výdajů.

Kapitola 3

Použité technologie

V této kapitole budou prezentovány informace o použitých technologiích při vývoji aplikace a důvody pro volbu daných technologií.

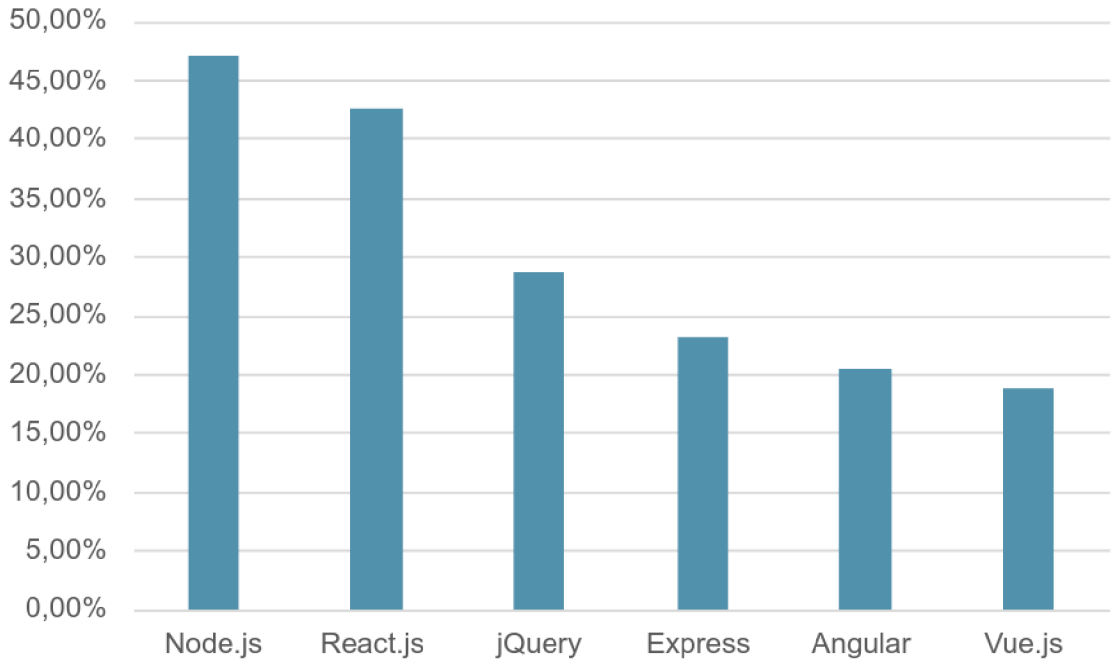
3.1 Webové frameworky

Dny, kdy se weby psaly pouze v HTML a CSS jsou již za námi. Ani s přidaným Javascriptem na rozpohybování HTML elementů, již dnes plnohodnotnou webovou aplikaci nenapíšeme jednoduše. Z webů se staly robustnější aplikace tzv. webové aplikace, které potřebují složitější logiku a tak vznikly webové frameworky.

Webové technologie jsou jedny z nejrychleji vyvíjejících technologií v oblasti informačních technologií. Webové aplikace jsou v podstatě aplikace na architektuře klient–server, kterou si popíšeme později. Běží ve webovém prohlížeči a komunikují přes protokol HTTP. Novým standardem pro webové aplikace se staly webové frameworky, které nám ulehčují jejich vývoj.

3.1.1 Single–page weby

Single–page aplikace je webová aplikace, jejíž veškerý obsah je tvořen jedinou stránkou. Uživatel této aplikace nemusí po každém kliknutí na nový obsah stránky stahovat celou stránku znovu, stačí pouze načíst obsah, který potřebujeme obnovit [14]. Single–page aplikace se v dnešní době dělají pomocí javascriptových frameworků a knihoven. Na obrázku 3.1 můžeme vidět 6 nejvyhledávanějších a nejvyužívanějších webových frameworků v roce 2022 [1]:



Obrázek 3.1: Popularita webových frameworků v roce 2022 [1].

Všechny si postupně rozebereme v následujících kapitolách.

3.1.2 Node.js

Node.js[1] umožňuje programátorům používat JavaScript na straně serveru, což znamená, že lze vytvářet komplexní webové aplikace pomocí jediného jazyka – JavaScriptu [16]. Node.js je ideální pro tvorbu rychlých, škálovatelných a vysoce výkonných webových aplikací, API nebo serverových služeb [15]. V předchozích letech Node.js nebyl považován za webový framework, ale v dnešní době ho již mnoho vývojařů považuje za plnohodnotný framework a tvoří základ pro vytváření dalších webových frameworků, jako například Express.js, který poskytuje další vrstvu abstrakce. V mé práci by volba Node.js dávala smysl pouze při volbě jazyka JavaScript pro serverovou část aplikace.

3.1.3 React.js

React.js[2] je javascriptová knihovna pro tvorbu uživatelského rozhraní. Vytvořena byla zaměstnancem firmy Facebook a je dále rozvíjena firmou Meta (dříve Facebook) [8]. Hlavní výhodou a také důvodem, proč v této práci bude použit React je virtuální DOM (Document Object Model), díky kterému může aplikace rychle reagovat na změny v datovém modelu a držet přitom stabilní výkon aplikace. Další skvělou výhodou je jedna z největších komunitních základen a rozsáhlá dokumentace. Správa stavů a synchronizace aplikace s datovým modelem je zde implementována přívětivě.

[1] https://nodejs.org/en
[2] https://react.dev/

3.1.4 jQuery

jQuery[3] je javascriptová knihovna s širokou podporou prohlížečů, která klade důraz na interakci mezi JavaScriptem a HTML. Byla vytvořena v roce 2006[4] a jedná se tak o nejstarší knihovnu v našem výběru a hlavně kvůli stáří ji nechci ve své práci používat.

3.1.5 Express

Express[5] je populární open–source framework pro Node.js, který umožňuje vytvářet webové aplikace a API. Express poskytuje jednoduché a minimalistické rozhraní pro vytváření serverových aplikací. Express je serverový javascriptový framework a protože bude náš server hodně pracovat s úpravou dat Javascript není úplně správná volba, proto ho nebudeme v naší práci používat.

3.1.6 Flask

Flask[6] je open–source webový framework pro programování webových aplikací v jazyce Python. Je to lehký framework, který nabízí jednoduchou a elegantní architekturu, což umožňuje programátorům rychle a snadno vytvářet webové aplikace. Mezi výhody použití Flasku patří jeho jednoduchá architektura, která umožňuje programátorům rychle se s ním seznámit a začít s vývojem webových aplikací. Flask také umožňuje rozšiřování a přizpůsobení podle potřeb vývojářů, což z něj činí vhodný framework pro malé a střední webové projekty. Flask také obsahuje rozsáhlou dokumentaci a silnou komunitu, což umožňuje vývojářům rychle najít řešení problémů. V naší aplikace bude Flask běžet na serveru a díky jazyku Python nám umožní zprovoznění umělé inteligence přes knihovnu PyTorch. Python je též skvělý a použitelný pro dolování z dat a práci s řetězcem textu.

3.1.7 Angular

Angular[7] je open–source webový framework vyvíjený společností Google, který slouží k vývoji moderních a dynamických webových aplikací. Angular využívá jazyk TypeScript, který umožňuje psát kód s větší jistotou, což zlepšuje jeho čitelnost a údržbu. Mezi nevýhody použití Angularu patří jeho složitější syntaxe, která může být pro programátory náročná. Angular také vyžaduje větší množství zbytečného kódu a konfigurace než jiné webové frameworky, což může vést k většímu objemu kódu a zvýšené složitosti aplikace.

3.1.8 Vue.js

Vue.js[8] je moderní a open–source frontendový framework pro vývoj webových aplikací a uživatelských rozhraní (UI). Byl vytvořen v roce 2014 a jeho hlavním tvůrcem je Evan You[9]. Jeho cílem je poskytnout jednoduchý a flexibilní framework pro vývoj moderních webových aplikací. Hlavní nevýhodou tohoto frameworku je malá komunita, což může vést k menšímu počtu návodů, knihoven a rozšíření.

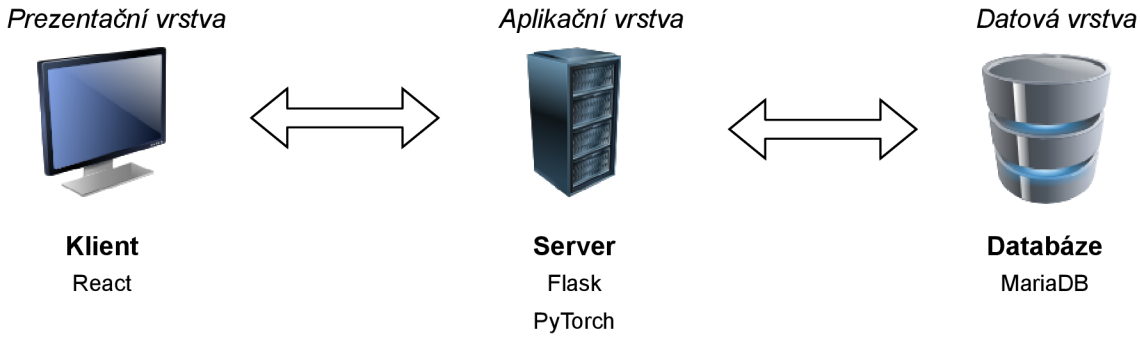
[3] https://jquery.com/
[4] https://blog.jquery.com/2006/08/26/jquery-10/
[5] https://expressjs.com/
[6] https://flask.palletsprojects.com/en/2.2.x/
[7] https://angular.io/
[8] https://vuejs.org/
[9] https://vuejs.org/about/faq.html

3.2 Architektura

Správná architektura webové aplikace může výrazně ovlivnit úspěch a celkovou kvalitu aplikace. Tím pádem je důležité pečlivě zvážit při návrhu architektury webové aplikace různé faktory jako jsou potřeby uživatelů, požadavky na výkon a škálovatelnost, zabezpečení a další. Celá architektura je postavena na konceptu Třívrstvé architektury. Následující kapitola je převzata z [21].

3.2.1 Třívrstvá architektura

Je to metoda rozdělení architektury aplikace do tří nezávislých vrstev – prezentační, aplikační a datová. Každá má svůj specifický úkol. Na obrázku 3.2 můžete vidět diagram propojení celé architektury.



Obrázek 3.2: Třívrstvá architektura.

Prezentační vrstva

Jediná vrstva se kterou interaguje uživatel, ten zadá vstupní data a zobrazí se mu výstupní data ve srozumitelné formě. Zahrnuje různé typy UI prvků, jako jsou formuláře, tlačítka, seznamy, tabulky a další. Tyto prvky umožňují uživatelům zadávat data, procházet seznamy a zobrazovat výsledky svých akcí. Prezentační vrstva má také vliv na celkový vzhled a pocit z aplikace. Vývojáři musí zvážit různé faktory, jako jsou typy uživatelů, účel aplikace, obecné návyky uživatelů a další faktory, při navrhování uživatelského rozhraní. V dnešní době se také klade důraz na přístupnost a zohlednění přístupu pro lidi s nějakým postižením. Prezentační vrstva je zastoupena v mé práci klientem a bude implementována frameworkem React.js.

Aplikační vrstva

Tato vrstva má na starosti logiku celé aplikace a stará se o zpracování dat. Přijímá požadavky prezentační vrstvy a ty dále zpracovává. Hlavním úkolem aplikační vrstvy je zpracování požadavků uživatelů. Aplikační vrstva je v mé práci zastoupena serverem a bude implementována frameworkem Flask. V této vrstvě bude probíhat kategorizace transakcí pomocí modelu umělé inteligence s knihovnou PyTorch.

Datová vrstva

Datová vrstva je nejnižší vrstva architektury, která se stará o ukládání a získávání dat z databáze. Hlavním úkolem datové vrstvy je zajistit, aby data byla ukládána bezpečně, konzistentně a efektivně, a aby byla snadno dostupná pro další zpracování v aplikaci. Správná funkčnost datové vrstvy je zásadní pro celkovou funkčnost aplikace, protože zajišťuje, že data jsou uložena a zpracována správným způsobem. V této vrstvě budu používat MariaDB.

3.2.2 Server

Na serveru budu využívat webový framework Flask, který nabízí celou řadu užitečných funkcionalit, včetně efektivní práce s textovými řetězci a schopnosti stavět API ve formě logických celků. Python disponuje rozsáhlou knihovnou funkcí a modulů jako např. knihovnu pro práci s umělou inteligencí PyTorch.

3.2.3 Klient

Pro klienta budu využívat webový framework React. Vybral jsem si jej zejména díky komunitní základně a využití virtuálního DOM modelu, který se bude hodit při tvorbě aplikace a zjednoduší některé činnosti. Díky propojení přes API nemusím řešit žádnou kompatibilitu mezi frameworky na serveru a klientu.

3.2.4 Databáze

Správná volba databázového systému je klíčová pro úspěšné dokončení projektu. MariaDB[10] je relační databázový systém, který vznikl v roce 2009 jako fork MySQL[11], a díky svým vlastnostem a výhodám si získal popularitu mezi uživateli i vývojáři. Je to open-source software s velkou komunitou uživatelů a vývojářů, což umožňuje uživatelům upravovat a rozšiřovat systém podle svých potřeb. MariaDB má silnou reputaci na trhu databázových systémů. MariaDB obsahuje několik vylepšených funkcí, které nejsou v MySQL k dispozici např. pokročilé dotazy, lepší indexování, optimalizaci dotazů, nové úložiště pro datové typy a další [3].

3.3 Síťová komunikace

V následující části budu popisovat, jakými protokoly a formáty server a klient komunikují mezi sebou. Přenos dat je šifrován přes HTTPS, data se posílají ve formátu JSON.

3.3.1 HTTP

HTTP (Hypertext Transfer Protocol) je protokol aplikační vrstvy. Dnes je hlavně používaný na přenos dat po síti. Základní princip tohoto protokolu je odesílání požadavku a očekávání odpovědi ze serveru. Protokol si neukládá jednotlivé požadavky, to značí že je bezstavový [12]. Bezstavovost budeme řešit pomocí autentizačních hlaviček s tokeny.

[10] https://mariadb.com/
[11] https://www.mysql.com/

Základní HTTP metody

HTTP metody jsou používány k určení, jakým způsobem bude zpracován požadavek odeslaný klientem na server. Při odesílání na adresu se přidávají k metodám také informace o verzi protokolu a různé hlavičky. Existují čtyři základní metody [10]:

- `GET` – Odesílá požadavek na získání informací na serveru.
- `POST` – Odesílá informace, které mají být zpracovány na serveru.
- `PUT` – Odesílá kompletní zdrojový soubor na server.
- `DELETE` – Odesílá informace, které se mají smazat na webovém serveru.

Dále existují také další metody, jako např. `HEAD`, `OPTIONS`, `TRACE` nebo `CONNECT`, které se používají v speciálních případech, v mé práci nebudou potřeba. Při přijímání odpovědi na požadavek dostáváme odpověď ve formě číselného kódu:

- 1xx – informační charakter,
- 2xx – operace schválena, akceptována,
- 3xx – operace přesměrování, potřeba dalších úprav pro splnění požadavku,
- 4xx – operaci nelze splnit,
- 5xx – chyba na straně serveru,

3.3.2 HTTPS

HTTPS (Hypertext Transfer Protocol Secure) je rozšíření již uvedeného aplikačního protokolu HTTP. Hlavní nevýhoda původního protokolu byla možnost sledování a čtení komunikace mezi serverem a klientem a takovou vlastnost u aplikace zaměřenou na finance nechceme. U HTTPS jsou použity kryptovací protokoly s asymetrickým šifrováním. Toto šifrování chrání komunikaci mezi klientem a server. Tato technologie zabraňuje podvržení zpráv i jejich čitelnost někým jiným než serverem a klientem, kteří mezi sebou tuto komunikaci vedou [20].

Application Programming Interface

API (Application Programming Interface) je komunikační rozhraní, které používají webové služby pro komunikaci mezi servrem a klientem. API jednoznačně odděluje prezentační část od aplikační a dovoluje napojit webovou i mobilní aplikaci pomocí stejného rozhraní. Mezi nejpoužívanější architekturu patří REST.

REST API

REST (Represational State Transfer) komunikuje prostřednictvím protokolu HTTP. Díky tomu může využívat již implementované hlavičky, metody, tělo a stavové kódy. Základní HTTP metody (GET, POST, PUT a DELETE) se přetransformují na CRUD (Create, Read, Update a Delete) operace, které zprostředkuje server [17]:

- `POST` → Create – Odesílá požadavek na vytvoření dat.

- `GET` → Read – Posílá zpátky informace o datech.
- `PUT` → Update – Odesílá požadavek na úpravu dat.
- `DELETE` → Delete – Odesílá požadavek na smazání dat.

REST architektura má obecně definované principy [17], které jsem se snažil dodržet v mé práci:

- Jednotné rozhraní
- Architektura klient–server
- Bezstavová logika
- Vrstvená architektura
- Kód na vyžádání
- Ukládání dat do cache

3.3.3 JSON

JSON (JavaScript Object Notation) je formát pro přenos dat. Jeho syntaxe je založena na JavaScriptových objektech a polích a umožňuje reprezentovat data ve formátu hierarchických stromů. Data jsou uloženy ve formátu `{"klíč":"hodnota"}`. Tento formát je čitelný jak pro stroje, tak i pro lidské oko [4].

3.4 Autentizace

Autentizace neboli ověření uživatelovy identity lze provádět mnoha způsoby. V mé aplikaci jsem zvolil klasický způsob ověření, a to přes uživatelův email a heslo. Tento email bude pro každého uživatele taktéž jedinečný identifikátor. Kromě tohoto klasického způsobu se stále více využívá biometrie, díky její rychlosti a přesnosti. U bankovních aplikací jsou také rozšířené PIN kódy, bezpečnostní kódy a nebo osobní otázky. V rámci zvýšení zabezpečení může společnost používat i více druhů autentizace a tak zvýšit bezpečnost na úkor uživatelské přívětivosti [6].

JWT token

JWT (JSON Web Token) je internetový standard pro formát tokenu, který lze použít pro autentizaci a autorizaci. Jedná se o kompaktní a samostatně přenositelný způsob přenosu informací o uživateli mezi různými částmi aplikace. JWT obsahuje zakódované informace o uživateli v JSON formátu, které jsou podepsané digitálním podpisem pro zajištění integrity dat [13]. Po autentizaci se vytvoří na serveru token (API endpointem `/token`), který pomocí API rozhraní přijme prohlížeč a uloží si ho do Local Storage. Local Storage (Web Storage) je jedna z funkcionalit moderních webových prohlížečů, která dovoluje aplikacím ukládat data na počítači uživatele. Tato data se ukládají přímo do paměti prohlížeče a jsou k dispozici i po restartu prohlížeče [23]. Při každém dalším volání API se do HTTP hlavičky přidá řádek s autentizačním tokenem. Tak server pozná o jakého uživatele se jedná a podle toho uzpůsobí dál svoje chování.

Kapitola 4

Analýza požadavků

V následující kapitole budou popsány jaké formáty bankovní společnosti nabízejí.

4.1 Formáty souborů bank

Při zkoumání výpisů formátu aplikace jsem kontaktoval všechny přední banky v České republice formou telefonického hovoru nebo emailem a zeptal jsem se, jaké formáty nabízí pro výpisy. Základním formát souboru je PDF, ten nabízí všechny bankovní společnosti. Tento formát se velmi špatně čte z pohledu automatizovaného procesu programem. Strojově čitelné formáty pro programy jsou třeba CSV nebo XML. Dle mého průzkumu minimálně jeden z těchto formátů poskytuje každá banka působící v ČR. Ze získaných informací jsem složil tabulku 4.1. V tabulce můžeme vidět, jaká banka které formáty používá.

Bankovní společnost	**CSV**	**CSV kódování**	**XML**	**XML kódování**	**PDF**
Moneta	✓	UTF–8 BOM	✓	UTF–8	✓
Komerční banka	✓	Windows–1250	×	×	✓
Equa	×	×	✓	UTF–8	✓
Creditas	✓	UTF–8	✓	UTF–8	✓
mBank	✓	Windows–1250	×	×	✓
Raiffeisenbank	✓	UTF–8	×	×	✓
AirBank	✓	Windows–1250	×	×	✓
ČSOB	×	×	✓	UTF–8	✓
Fio	✓	UTF–8	×	×	✓
UniCredit bank	✓	Windows–1250	×	×	✓

Tabulka 4.1: Formáty souborů bank.

V následujících kapitolách si stručně popíšeme jednotlivé formáty a představíme, která data budeme využívat.

Formát PDF

Portable Document Format je formát souboru pro elektronické uložení dokumentu, který byl vyvinut společností Adobe. Je to jeden z nejpopulárnější formátů pro dokumenty, protože dokáže zobrazit a uložit dokument v podobě, ve které bude vypadat stejně nezávisle

na softwaru a hardwaru, kde se prohlíží nebo vytváří. PDF dokumenty jsou často používány pro publikace, prezentace, formuláře a další dokumenty, které mají být zachovány v původním formátu, bez nutnosti úprav. PDF dokumenty lze zobrazovat a tisknout na různých zařízeních, včetně počítačů, tabletů a chytrých telefonů. PDF formát je otevřený standard a existuje mnoho různých nástrojů pro tvorbu, úpravu a zobrazování PDF dokumentů [2]. Strojové čtení PDF dokumentů může být náročné, protože PDF není primárně určen pro strojové zpracování. PDF dokumenty jsou v podstatě digitální obrazy původních dokumentů, které obsahují text, obrázky a další prvky, a ty jsou uloženy v grafickém formátu. Existuji technologie, které dokážou strojově číst tyto soubory jako třeba OCR (Optical Character Recognition), ale přesné rozpoznávání znaků může být problematické, pokud jsou v PDF dokumentu použity různé fonty, velikosti písma nebo pokud jsou texty zakřivené.

Formát XML

Extensible Markup Language (zkráceně XML) je formát souboru pro značkovací zápis (zápis obohacený o informace o struktuře, významu a zobrazení textu). Vytvořený mezinárodní konsorciem W3C[1]. XML umožňuje strukturovat data pomocí značek a atributů, podobně jako HTML. Detailní specifikace tohoto formátu je dostupná z [9]. Nicméně na rozdíl od HTML je XML univerzálnější a může být použit pro různé typy dat. Neexistuje žádná standardizace mezi bankami a tak používá každá banka jiné zápisy. XML jde velmi pěkně strojově číst, proto ho využijeme v naší práci.
Příklad výpisu jedné transakce od Monety:

```
<transaction id="220610V0385345" other-account-number="PLATBA KARTOU" date-
    post="2022-06-09" date-eff="2022-06-13" var-sym="4152280774" con-sym="
    1178" spec-sym="" amount="-24.90" contactless="0" mobile-payment="0">
        <trn-messages type="description">
                <trn-message position="2">ALBERT VAM DEKUJE BRNO CZ</trn-
                    message>
        </trn-messages>
</transaction>
```

Příklad výpisu jedné transakce od ČSOB (zkráceno):

```
<Ntry>
    <NtryRef>20503</NtryRef>
    <Amt Ccy="CZK">154.90</Amt>
    <CdtDbtInd>DBIT</CdtDbtInd>
    <RvslInd>false</RvslInd>
    <Sts>BOOK</Sts>
    <BookgDt>
        <Dt>2023-01-11</Dt>
        </BookgDt>
        <BkTxCd>
                <Prtry>
                        <Cd>30000301000</Cd>
                        <Issr>Czech Banking Association</Issr>
                </Prtry>
```

[1] https://www.w3.org/

```
        </BkTxCd>
        <NtryDtls>
                <TxDtls>
                <Refs><AcctSvcrRef>20503</AcctSvcrRef></Refs>
                        <BkTxCd>
                        <Prtry><Cd>30000301000</Cd>
                        <Issr>Czech Banking Association</Issr></Prtry>
                        </BkTxCd>
                <RmtInf> <Ustrd>Misto: Penny Mocidla 2369 Uhersky Brod
                    Castka: 154.9 CZK 09.01.2023</Ustrd>
                </RmtInf>
                </TxDtls>
    </NtryDtls>
</Ntry>
```

Formát CSV

Comma–Separated Values (zkráceně CSV) je formát souboru pro ukládání tabulkových dat. Každý řádek v souboru reprezentuje jeden záznam a každý sloupec představuje jednu hodnotu v záznamu. Data ve sloupci jsou oddělená oddělovačem. Jako oddělovač je nejčastěji používána čárka nebo středník. CSV formát je velmi jednoduchý a snadno čitelný pro stroje i lidi, což ho činí ideálním pro výměnu dat mezi různými aplikacemi a systémy. Detailní specifikace tohoto formátu je dostupná z [22]. CSV formát není v bankovnictví standardizovaný a může se lišit v závislosti na společnosti, která ho používá.
Příklad výpisu Komerční banky (zkráceno):

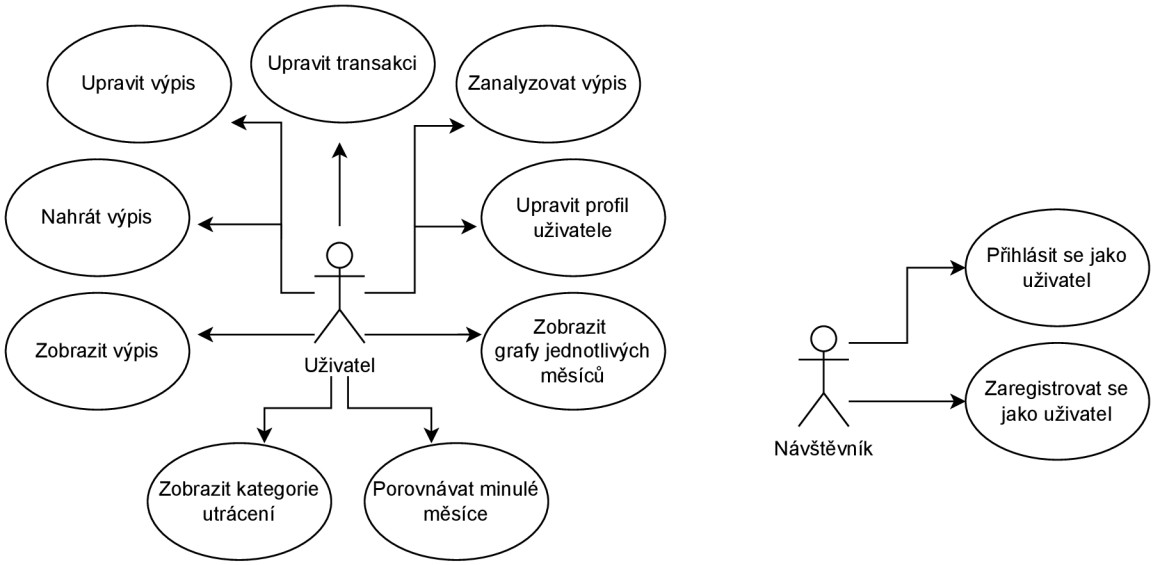
```
"Datum splatnosti";"Datum odepsání z jiné banky";"Protiúčet a kód banky";
"Název protiúčtu";"Částka";"Originální částka";"Originální měna";"Kurz";
"VS";"KS";"SS";"Identifikace transakce";"Systémový popis";"Popis příkazce";
"Popis pro příjemce";"AV pole 1";"AV pole 2";"AV pole 3";"AV pole 4";
"21.11.2022";;"123-3114990237/0100";"PRIME VISA - PLATEBNÍ KARTY CZ";
"-448,00";"";"";"";"10310525";"1178";"102112001";
"244-21112022 10860416769942";"Nákup na internetu ";
"0416769942 0416769942";"0010310525    000000";
"GOPAY  *GACINEMA.CZ";"gacinema.cz
```

Příklad výpisu Fio banky:

```
"ID operace";"Datum";"Objem";"Měna";"Protiúčet";"Název protiúčtu";
"Kód banky";"Název banky";"KS";"VS";"SS";"Poznámka";
"Zpráva pro příjemce";"Typ";"Provedl";"Upřesnění";"Poznámka";
"BIC";"ID pokynu"
"24022016996";"01.02.2022";"-3500";"CZK";"35-1323700217";"";"0100";
"Komerční banka a.s.";"";"";"";"";"";"Bezhotovostní platba";
"";"";"";"";"30251738792"
```

4.2 Diagram případů užití

Diagram případů užití zachycuje vnější pohled na modelovaný systém. Modeluje hlavně hranice systému a požadované aktivity aktérů.



Obrázek 4.1: Diagram případů užití.

Na obrázku 4.1 můžeme vidět vymodelovaný diagram pro náš navrhovaný systém. Bude mít dva aktéry uživatele a návštěvníka.

Uživatel

Uživatelova hlavní akce bude nahrát výpis. Při nahrání výpisu uživatel také může nechat výpis zanalyzovat umělou inteligencí. Od této akce se odvíjí celá řada dalších akcí. Bude mít tyto možnosti: porovnávat jednotlivé měsíce, zobrazit kategorie utrácení, zobrazit daný výpis, upravit jednotlivé transakce, upravit výpis a upravit profil.

Návštěvník

Návštěvník aplikace, kterého potřebujeme rozpoznat a zajistit mu jednoznačný identifikátor. Tento návštěvník má možnost se zaregistrovat nebo přihlásit se a tím se přepnout na aktéra uživatel.

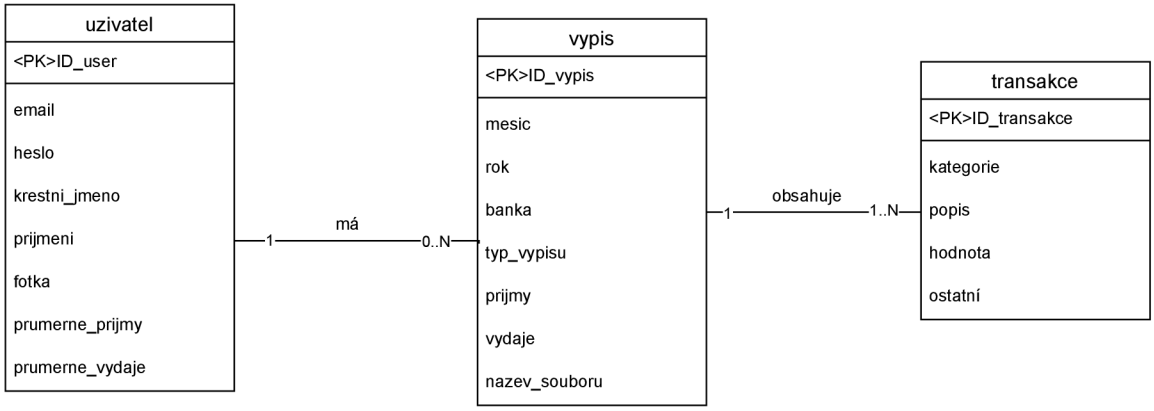
Kapitola 5

Návrh

V této kapitole bude popsán návrh jednotlivých částí aplikace.

5.1 Entity–relationship model

ER model se v softwarovém inženýrství používá pro abstraktní a konceptuální znázornění dat. Znázorňuje, jak bude vypadat naše rozložení dat v databázi a jejich vzájemné vztahy [18].

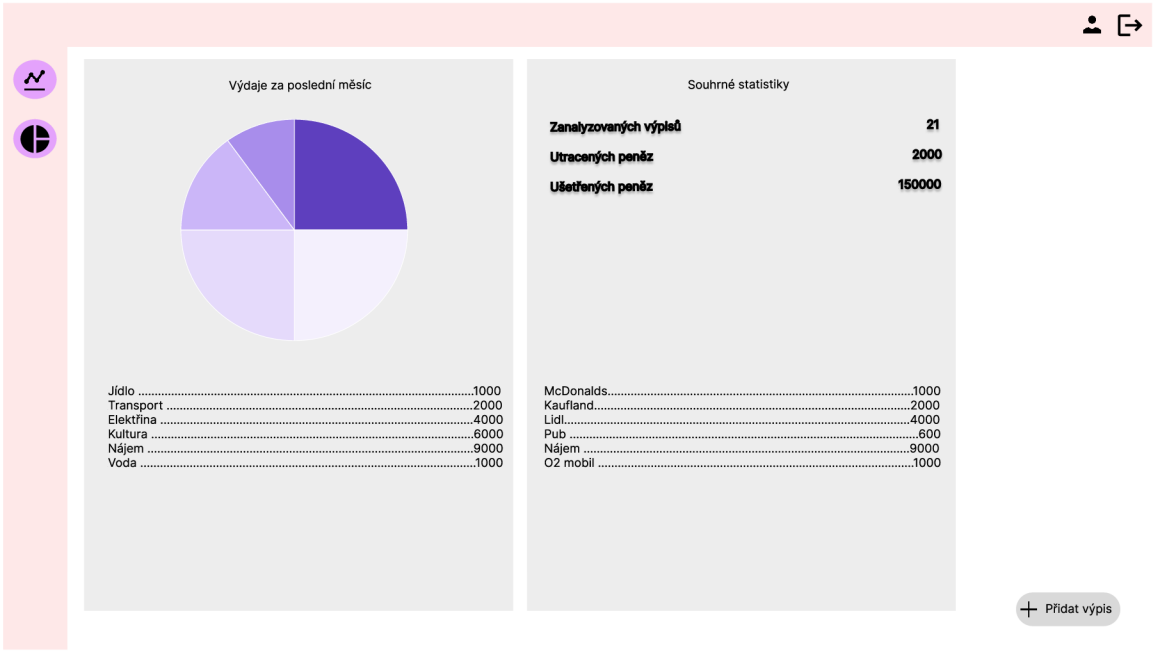


Obrázek 5.1: ER diagram.

Náš diagram (obrázek 5.1) bude mít tři entitní množiny – uživatel, výpis a transakce. Uživatel bude mít atributy id, email, heslo, křestní jméno, příjmení, fotka, průměrné příjmy a průměrné výdaje. Výpis bude mít atributy id, měsíc, rok, banku, typ výpisu, příjmy, výdaje a název souboru. Transakce bude mít atributy id, kategorie, popis, hodnota a ostatní.

5.2 Návrh uživatelského rozhraní

Navrhování uživatelského rozhraní (UI design) se v dnešní digitální době stalo nezbytnou součástí tvorby jakékoliv webové aplikace. Cílem je, aby uživatelé byli schopni snadno ovládat produkt, najít informace a vykonat požadované úkoly bez zbytečného stresu nebo frustrace. Při mém prvotním navrhováním, jak bude mít aplikace rozmístěné UI prvky jsem zvolil následující rozvržení (obrázek 5.2).



Obrázek 5.2: Wireframe domovské obrazovky.

Hlavní a „úvodní“ obrazovka, kterou uživatel uvidí první po přihlášení, bude obrazovka s analýzou výpisu z posledního měsíce. Nejsignifikantnějším prvkem bude graf s barevně rozlišenými kategoriemi. Tady si po pravé straně uživatel bude moct prohlížet a upravovat transakce z posledního měsíce. Uživatel bude moci najet myší na graf a zobrazit přesně, ke které barvě patří jaká kategorie a kolik v ní utratil. Navigace bude mít 5 nejdůležitějších tlačítek Domů, Výpisy, Grafy, Profil a Odhlášení. Nejdůležitější tlačítko volající o akci uživatele bude v levém dolním rohu `+ Přidat výpis`. Stisknutím tlačítka uživatel nahraje výpis a následně spustí analýzu transakcí. To vše v modálním okně, které bude taky znovu použitelné pro editace transakcí nebo výpisů. Kde ve formuláři bude každé políčko jeden atribut z ER Diagramu (obrázek 5.1) a půjde jednoduše upravovat.

Na druhé hlavní obrazovce se statistikou může uživatel pozorovat své finance v dlouhodobém měřítku. Může usoudit, ve které oblasti je potřeba zlepšit své osobní finanční rozhodování. Bude mít k dispozici souhrnné grafy za posledních několik měsíců, jak lze vidět na obrázku 5.3.

Obrázek 5.3: Wireframe obrazovky se statistikou.

5.3 Návrh API

Správný návrh API je klíčovým prvkem při vývoji aplikace a je nutné se mu věnovat s dostatečnou pozorností a pečlivosti. Základní předpoklad pro návrh API je analýza návrhu UI. Zde je potřeba popřemýšlet, jaké datové podklady na které stránce jsou potřeba. Pro každou entitu je potřeba navrhnout 4 základní HTTP metody. Tyto metody nám poslouží k vytvoření nové entity, získání dat o entitě, úpravu entity a smazání entity. Tento druh endpointů je základ kvalitní aplikace. Seznam navržených endpointů na serveru, které vyvolávají akci:

- `GET, POST /user`
- `PUT, DELETE /user/<user_id>`
- `POST /statement`
- `GET, PUT, DELETE /statement/<statement_id>`
- `POST /transaction`
- `GET, PUT, DELETE /transaction/<trans_id>`
- `POST /token`
- `GET /lateststatement`
- `GET /allstatements/`
- `GET /statement/<statement_id>/transactions`
- `GET /analyze/<statement_id>`

Pojmenování těchto CRUD endpointů se řídí pravidly REST API. To znamená, že endpointy (transakce, výpis) se pojmenovávají dle tohoto klíče. GET, PUT a DELETE `/<jméno_entity>/<identifikační_číslo>` a POST `/<jméno_entity>/`. Na výpisu 5.1 lze vidět jak požadavek a odpověď vypadají.

```
GET http://127.0.0.1:5000/statement/84
{
    "bank": "Komercka",
    "filename": "kb_november.csv",
    "id": 84,
    "income": 40462,
    "month": 12,
    "spending": 40090,
    "type": "Checkings",
    "year": 2022
}
```

Výpis 5.1: Příklad požadavku `GET /statement/84`.

Pojmenování endopointu pro entitu uživatel je odlišné, protože díky JWT autentizačnímu tokenu v hlavičce požadavku dokážeme rozeznat uživatele bez potřeby identifikačního čísla umístěného na konci endpointu. Z tohoto důvodu bude pro metody GET a POST endpoint pojmenován `/uzivatel/` a pro PUT a DELETE bude pojmenován `/uzivatel/<identifikační_číslo>`.

Další endpoint `/token` slouží na získání autentizačního tokenu JWT, který následně vloží klient do hlavičky HTTP požadavku. Tento endpoint je důležitý pro přihlášení a autentizaci uživatele.

Následuje skupina endpointů `/lateststatement`, který vrací výpis z posledního měsíce a `/allstatements`, který vrací všechny uživatelovi výpisy, zjištění uživatele lze znova poznat pomocí autentizační hlavičky. Kombinovaný endpoint dvou entit `/statement/<statement_id>/transactions` vrací data o všech transakcích, které daný výpis obsahuje.

Poslední endpoint `/analyze` nevrací nic, pouze provede na daném výpisu kategorizaci transakcí a přidá transakce do databáze.

Kapitola 6

Implementace

V následující kapitole budou popsány implementační detaily serveru a klienta.

6.1 Implementace serveru

Server běží na webovém frameworku Flask. Server je rozdělen do několika podsouborů. Hlavní soubor, který tvoří jádro celé aplikace je pojmenovaný `server.py`, zde běží celé API, a v dalších třech souborech běží model umělé inteligence (`model.py`), extrakce dat z výpisu (`template_analyze.py`) a kategorizace výpisů bez AI (`static_analyze.py`).

6.1.1 Implementace API

Flask nabízí jednoduché a minimalistické vytvoření API. K rozběhání serveru stačí inicializovat Flask do proměnné `app` a následně ji spustit. Pro napojení na databázi potřebujeme Python knihovnu SQLAlchemy[1]. Tato knihovna nám umožní pracovat s databází v prostředí Python a zpřístupní nám možnost vymodelovat databázi. Podobně jako Flask i SQLAlchemy se musí inicializovat a následně pracovat pod proměnou `db`. Pro každou entitu z ER diagramu vytvoříme třídu s definicí, jaké atributy má daná třída mít v databázi. Pro získávání dat z databáze pomocí dotazů je vytvořena speciální třída se schématem, který je typově stejný jako související třída pro databázový model. Tato třída je z knihovny Marshmallow[2] slouží pro deserializaci databázového objektu. To znamená, že díky převodu na schéma je možné rovnou posílat data v JSON formátu.

Každý endpoint se definuje ve webovém frameworku Flask pomocí `@app.route('/cesta', methods=["POST"])`, zde si může nastavit požadovanou URI a povolené HTTP metody.

V metodě GET pomocí knihovny SQLAlchemy je vytvořen hledací dotaz do databáze (např. `User.query.filter_by(id=user_id).first()`). Tento příklad vrací objekt User s hledaným identifikátorem, díky převodu na schéma můžou být informace o objektu poslány zpět na klienta.

V metodě POST se odeslaná data z klienta na serveru přijmou ve formátu JSON. Na serveru se uloží do proměnné pomocí `request.json` všechny informace z těla HTTP zprávy. Tato proměnná použije funkci `data.get('klíč')`. Klíč definuje, která hodnota z JSON formátu bude vybrána a uložena. Poté se všechny hodnoty uloží do třídy s příslušným názvem a pomocí `db.session.add('třída')` a `db.commit()` se objekt vkládá do databáze.

[1] https://www.sqlalchemy.org/
[2] https://flask-marshmallow.readthedocs.io/en/latest/

Metoda PUT operuje podobně jako metoda POST s tím rozdílem, že se první pokusí najít objekt, který má upravovat v databázi.

Metoda DELETE pouze vyhledá v databázi objekt a příkazem `db.session.delete(objekt)` smaže objekt. Po každé úpravě databáze musí být volána funkce `db.session.commit()` pro perzistentní stav databáze.

6.1.2 Extrakce dat z šablony

Jak jsem již formuloval v kapitole 4.1, každá banka má rozdílné výpisy transakcí. Proto je potřeba pro každou banku vytvořit vlastní funkci k extrakci dat.

AirBank

AirBank používá formát CSV v kódování WIN-1250. Funkce `Airbank_analyze()` otevře soubor pro čtení a pomocí knihovny pro čtení CSV souboru otevře soubor a hledá požadované sloupečky v souboru: „Název protistrany“ a „Částka v měně účtu“ tyto dvě informace nám stačí pro vytvoření objektu Transakce a spuštění automatické klasifikace transakce.

Fio

Fio nabízí své výpisy v CSV formátu. Postup je podobný jako u AirBank. Otevřeme CSV soubor a hledáme sloupečky v souboru, které obsahují podrobné informace o transakci („Název obchodníka“ popřípadě „Poznámka“, „Zaúčtovaná částka“).

Československá obchodní banka

ČSOB nabízí výpisy ve formátu XML i s jejich detailně popsanou dokumentací[3]. Pomocí knihovny XML pro načtení stromové struktury do proměnné postupně procházíme strukturou a hledáme položky s tagem „Ntry“ pro zjištění, že se jedná o záznam o transakci. V transakci hledáme „Amt“ pro zjištění částky a dále zjišťujeme pomocí tagu „CdtDbtInd“ zda je platba kladná („CRDT“) nebo záporná („DBIT“). A po dalším zanoření hledáme „NtryDtls“ neboli detail transakce. A pro samotný popis transakce se musíme zanořit ještě o dvě úrovně níž na položku „Ustrd“, ze které musíme odstranit lokaci a částku pro další kategorizaci.

Moneta

Moneta taktéž nabízí výpisy ve formátu XML a poskytuje jejich detailně popsanou strukturu[4]. Moneta má na rozdíl od ČSOB jednodušší stromovou strukturu a hledáme pouze elementy s tagem „transaction“, kde pak hledáme atributy s názvem „amount“ pro částku a pro popis transakce se zanoříme do „trn-messages“ a následně do „trn-message“. V tomto bodě zanoření byl nalezen popis transakce a můžeme spustit automatickou kategorizaci a uložit objekt do databáze.

Komerční banka

Komerční banka poskytuje své výpisy ve formátu CSV. Na rozdíl od ostatních bank přidává do svých výpisů nadstandardně velkou hlavičku, kterou je nutné odstranit. Po analýze

[3] https://www.csob.cz/portal/documents/10710/1927786/ceb-vypisy-format-xml.pdf
[4] https://www.moneta.cz/documents/20143/11740743/mmb-ib-struktura-vypisu-xml.zip

několika výpisů od různých lidí se mi potvrdilo, že nejlepší cestou bude otevřít soubor odstranit z něj prvních 17 řádků, uložit a až potom ho načíst knihovnou pro čtení CSV. Dále procházíme soubor a hledáme sloupečky „AV pole 1“ a pole „Částka“. Tato data nám stačí pro přidání objektu Transakce do databáze. Zkusíme spustit automatickou detekci kategorie a transakci uložíme.

UniCredit bank

UniCredit banka poskytuje výpisy ve formátu CSV. Tyto výpisy podobně jako výpisy z Komerční banky obsahují hlavičku. Proto je nutné odstranit z nich prvních 5 řádků, uložit soubor a až potom jej načíst knihovnou pro čtení CSV. Dále procházíme soubor a hledáme sloupce „Částka“ a „Příjemce“. Následně spustíme automatické řazení transakcí a uložíme transakce do databáze.

Budoucí přidání dalších bank

Budoucí rozvoj aplikace a přidávání všech českých a světových bankovních společností je vítaným rozšířením. Největším problémem s přidáním nové banky je získání dostatečného množství vzorků výpisů z banky. Bankovní společnosti nebyly nejochotnější v komunikaci s poskytnutí testovacích výpisů. Po dostatečné analýze je potřeba pouze vytvořit novou funkci v souboru `template_analyze.py`. Je potřeba vytvořit funkci, která přečte celý soubor, najde příslušné položky ve výpisy (částku, popis transakce), provede kategorizací a přidá transakci do databáze.

6.2 Implementace klienta

Klient běží na webovém frameworku React. Celkové rozložení implementace je složeno z menších komponent, které dohromady tvoří celé stránky v aplikaci.

6.2.1 Použité balíčky

React má obrovskou komunitní základnu a proto je k dispozici mnoho balíčků, které nám s ním umožní lépe pracovat. V následujících kapitolách popíši jen ty nejzajímavější.

Material UI

Material UI[5] je open source knihovna uživatelského rozhraní pro React framework, která poskytuje předdefinované komponenty a stylizaci v souladu s designovým jazykem Material Design od Google[6]. Material UI přináší rozsáhlou nabídku přizpůsobitelných komponent, zahrnujících tlačítka, formulářová pole, ikony, nabídky a další prvky, které lze snadno implementovat pro vytvoření moderního, jednotného a esteticky příjemného uživatelského rozhraní. Tato knihovna nabízí možnosti vysokého stupně flexibility a přizpůsobení, což umožňuje vývojářům snadno upravit vzhled a funkčnost komponent, které zrovna vývojář potřebuje. Knihovna disponuje skvěle zpracovanou dokumentací, což představuje jednu z hlavních faktorů, proč jsem se rozhodl ji využít.

[5] https://mui.com/
[6] https://m1.material.io/

Axios

Axios[7] je Javascriptová knihovna, která umožňuje vytváření HTTP požadavků v prostředích s použitím React frameworku. Tato knihovna poskytuje vývojářům širokou knihovnu funkcí pro získávání a odesílání dat mezi klientem a serverem.

Toastify

Toastify[8] je knihovna pro React, která umožňuje uživatele upozorňovat (tzv. toast zprávou) na různé změny a události. Tyto upozornění se většinou nachází v jednom ze čtyř rohů obrazovky.

Recharts

Knihovna Recharts[9] je open–source a slouží k vytváření interaktivních grafů. Knihovna nabízí mnoho různých typů grafů, jako jsou liniové grafy, sloupcové grafy, koláčové grafy, časové osy a další. Stala se velmi populární hlavně z důvodů obrovské možnosti přizpůsobení grafů (např. barvy, šířku čar, styl os atd.).

6.2.2 Komponenty

Dle návrhu je práce rozdělena na několik menších stránek. Každá stránka se skládá ze samostatné komponenty nebo z více komponent. Ve složce `src/componenets` je pro každou komponentu vyhrazen samostatný soubor. Zajímavější komponenty jsem se snažil rozebrat v následujících kapitolách.

App.js

Funkce `App` definuje a inicializuje základní strukturu aplikace. Používá se zde komponenta BrowserRouter, která poskytuje routing pro aplikaci, a inicializuje komponentu ToastContainer, která bude zobrazovat zprávy pro uživatele v rámci celého prostředí aplikace. Routing pro aplikaci znamená, že vytvoří cesty pro jednotlivé stránky, aby měla každá stránka svou URL.

UseToken.js

Tato komponenta pracuje hlavně s JWT tokenem. Obsahuje tři funkce `getToken`, `saveToken` a `removeToken`. Funkce `saveToken` uloží JWT token do Local Storage a pokud bude klient posílat další požadavky bude token jednoduše získatelný. Funkce `getToken` dokáže získat uložený JWT token z Local Storage a znovu ho použít při dalším HTTP požadavku. Funkce `removeToken` odstraní z Local Storage token a tím se uživatel nemá jak autentizovat při dalším HTTP požadavku.

Login.js

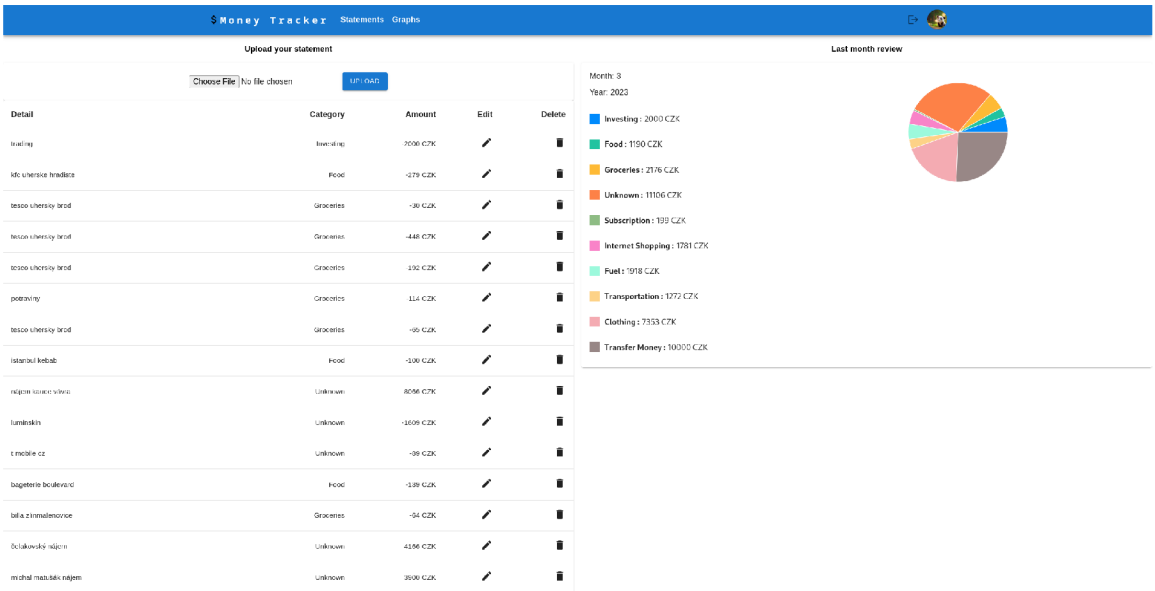
Je jedna z nejdůležitějších komponent aplikace, zde se uživatel autentizuje do aplikace. Uživatelovi se zobrazí formulář pro zadání svých inicializačních údajů. Pomocí zadání své

[7] https://www.npmjs.com/package/axios
[8] https://fkhadra.github.io/react-toastify/introduction
[9] https://recharts.org/en-US/

emailové adresy a hesla, a následným kliknutím na tlačítko `Sign in`, odešle klient pomocí Axios knihovny HTTP požadavek, který vrátí autentizační token. Pokud server vrátí úspěšnou odpověď na požadavek, přesměruje se klient na stránku `Home.js`. V případě, že autentizace selže, dojde k resetování polí pro zadání údajů a zobrazení zprávy o neúspěchu.

Home.js

Domovská obrazovka, jak z návrhu vyplývá, je první obrazovka se kterou se setká uživatel. Vnitřní logika volá endpoint pro získání dat z posledního výpisu a následuje endpoint na všechny transakce daného výpisu. Tato data dále posílá jako argumenty do dalších komponent. Na obrázku 6.1 lze vidět rozdělení na 4 komponenty – Navigační lišta (`NavBar`), Nahrávání výpisu (`Upload`), Výpis transakcí (`Transaction`) a Výsečový diagram (`Graph`).



Obrázek 6.1: Domovská obrazovka aplikace.

Transaction.js

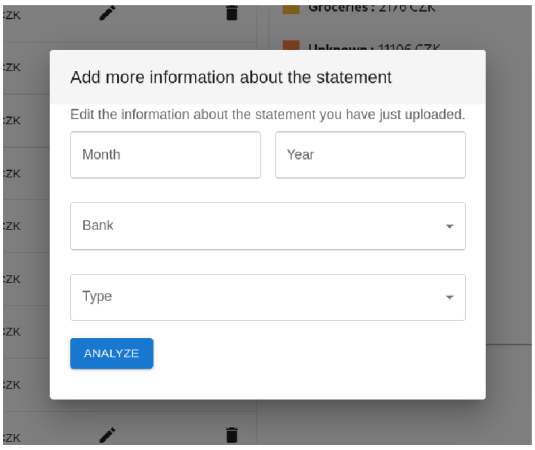
Transactions zobrazuje tabulku všech transakcí, které jsou předány jako argument. Pomocí mapovací funkce `transaction.map()` se tabulka rozšíří nebo zmenší na základě poskytnutých dat. Tabulka zobrazí všechny důležité parametry entity Transakce. Uživatel může díky ikonám rychle upravovat jednotlivé transakce v modální okně nebo je rovnou smazat.

Graph.js

Komponenta obsahuje funkci `countAmountByCategory`, která vezme data z transakcí a spočítá sumu za jednotlivé kategorie, které se nachází v datech. Funkce data připraví do formátu `kategorie:částka`, aby je mohla knihovna `Recharts` použít. Tato data se potom zobrazí v barevném grafu. Tato komponenta se zobrazuje Graf na základě dat transakcí, které komponenta dostane jako argument.

Upload.js

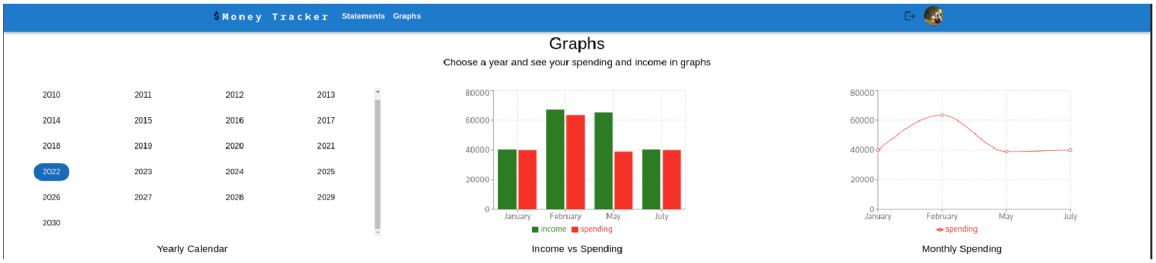
Tato komponenta dovolí uživatelovi nahrát soubor. Po kliknutí na tlačítko `Upload` se soubor pomocí POST požadavku odešle na server a pokud přijde kladná odpověď, klient si uloží identifikační číslo výpisu. Uživatelovi se zobrazí modální okno s požadavkem na vyplnění dodatečných údajů, které se pomocí PUT požadavku pošlou zpět na serveru a začne analýza transakcí. Dokud není hotova uživatel vidí načítací kolečko, jakmile klient přijme kladnou odpověď serveru přesměruje klienta na detail výpisu.



Obrázek 6.2: Modální okno pro doplnění informací o výpisu.

Graphs.js

Stránka s přehlednými grafy je schovaná v komponentě `Graphs.js`. Zde se kromě Navigační lišty nachází dva grafy, jeden zobrazující porovnání příjmů a výdajů v průběhů měsíců v daném roce. Na druhém grafu se zobrazuje křivka výdajů. Pomocí komponenty YearCalendar, která při změně vybraného roku vždy pošle nový požadavek na server a znovu načte dotčené komponenty, může uživatel měnit časové rozmezí pro data v grafu.

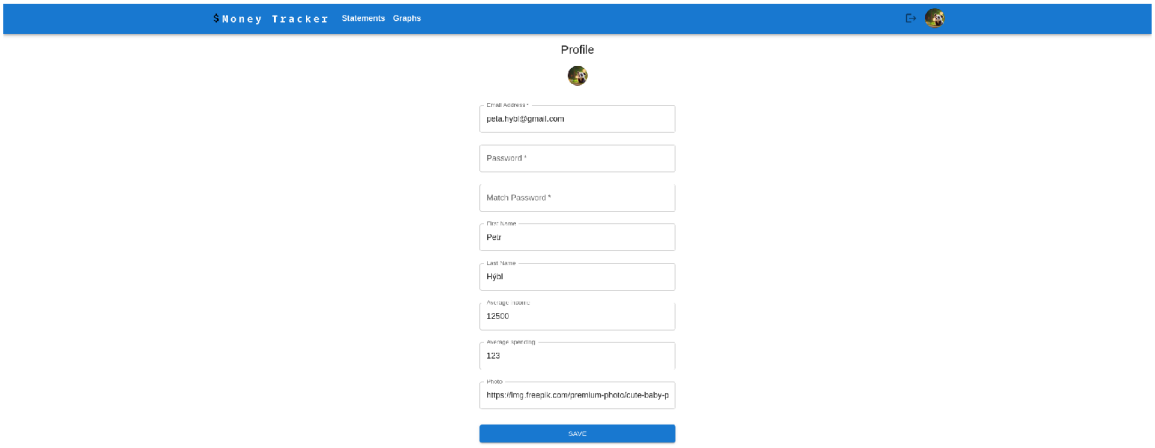


Obrázek 6.3: Obrazovka pro sledování výdajů v čase.

Profile.js

Tato stránka umožní uživateli upravit svůj profil, pomocí komponenty TextField z Material UI. Předtím než uživatel bude moct data upravovat je zavolán endpoint `GET /user`, ten nám zajistí nahrání dat do formulářových polí. Uživatel může upravovovat všechny položky, které obsahuje entita Uživatel. Tlačítko vyvolá akci `SubmitPut`, tato akce zkontroluje zda heslo bylo upraveno a zkontroluje zda bylo zadáno dvakrát. Tato kontrola je dů-

ležitá z důvodu velice častých chyb, které může uživatel udělat. Následně odešle požadavek `PUT /user/<identifikační_číslo>` a klient zobrazí dle odpovědi serveru kladnou nebo zápornou zprávu. Na obrázku 6.4 lze vidět úpravu jednoho z uživatelů.



Obrázek 6.4: Obrazovka pro úpravu uživatele.

Statements.js

Uživatel využije tuto komponentu při zobrazování detailu jednoho určitého výpisu. Komponenta načítá data z endpointu `/allstatements/`. Po načtení dat jsou zobrazeny v tabulce, kde každý řádek odpovídá jednomu bankovnímu výpisu a zahrnuje informace o měsíci, roce, bance, typu, příjmu a výdajích. Kliknutím na tlačítko `Edit` v řádku daného bankovního výpisu se otevře dialogové okno s editačním formulářem, který umožňuje změnit údaje v daném záznamu. Po změně údajů je možné uložit změny na server pomocí tlačítka, které volá na endpoint `/statement/<identifikační_číslo>`.

Kapitola 7

Automatická klasifikace transakcí

Různé banky nabízí různé formáty výpisů, jak jsem již uvedl v kapitole 4.1, a tak pokud je chceme automaticky kategorizovat, je potřeba extrahovat stejná data z každého výpisu. Nejdůležitějším parametrem je „Popis transakce“ (v každé bankovní společnosti se tento parametr nazývá jinak např. „Zpráva“ nebo „AV Pole 1“ atd.). Tento parametr poskytuje mnoho užitečných informací, jako je jméno příjemce nebo název společnosti, který lze využít pro další kategorizaci.

7.1 Python knihovny

Automatická kategorizace pomocí umělé inteligence se neobejde bez doplňujících knihoven. V následujících kapitolách si některé upřesníme k čemu budou použity.

7.1.1 PyTorch

PyTorch[1] je framework pro strojové učení, který byl vyvinut především pro výzkum v oblasti umělé inteligence a neuronových sítí. Tato knihovna umožňuje vývojářům vytvářet a trénovat složité neuronové sítě. Tuto knihovnu jsem si zvolil pro její vysokou flexibilitu a efektivitu. PyTorch je základem pro další knihovny, které dále budu používat.

7.1.2 Sentence Transformer

Sentence Transformer je nadstavba pro framework PyTorch, která dokáže vytvořit reprezentaci vět, textů a obrázků. Tento nástroj lze použít pro výpočet vektorů vět pro více než 100 jazyků, a poté pomocí kosinové podobnosti porovnat věty s podobným významem. Tato funkce je užitečná pro sémantické hledání podobností, sémantické vyhledávání nebo hledání parafrází. Nabízí spoustu předtrénovaných modelů a pro moji práci zvolíme model PARAPHRASE-MULTILINGUAL-MPNET-BASE-V2, který byl vytrénován na vícejazyčném datasetu a je tak vhodný i na práci s češtinou [19]. Tento model slouží k vytvoření vektorových reprezentací textu pomocí metody zvané „sentence embedding“. Tato metoda umožňuje převést celou větu nebo odstavec na jednorozměrný vektor, který reprezentuje její sémantický obsah. Tento vektor obsahuje informace o významu a kontextu celé věty.

[1] https://pytorch.org/

Kosinová podobnost

Kosinová podobnost je míra podobnosti dvou vektorů, která se získá výpočtem kosinu úhlu těchto vektorů [11]. Tímto vektorovým porovnání dokážeme rozeznat podobnost jednoho textu k dalšímu.

7.1.3 Pandas

Pandas[2] je knihovna pro jazyk Python, která umožňuje efektivní zpracování dat. Knihovna pandas nabízí nástroje pro načítání, manipulaci a analýzu dat v různých formátech, včetně CSV, Excel, SQL databází a mnoha dalších. Tato knihovna nám ulehčí práci při načítání datasetu pro trénování umělé inteligence.

7.1.4 Natural Language Toolkit

Natural Language Toolkit[3] (NLTK) je knihovna pro jazyk Python, která umožňuje snadné zpracování a analýzu přirozeného jazyka. NLTK poskytuje nástroje pro tokenizaci, stemming, lemmatizaci, syntaktickou analýzu a mnoho dalších úloh v oblasti zpracování textu. V mé práci ji hlavně využijeme na tokenizaci textu.

7.1.5 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) je jazykový model pro strojové učení, který vytvořila společnost Google. Jedná se o naučený model založený na architektuře transformátorů, který se specializuje na zpracování přirozeného jazyka. BERT je schopen provádět širokou škálu úloh, jako je například klasifikace textu, zodpovídání otázek, generování textu a mnoho dalších [7]. BERT s knihovnou Sentence Transformer budou posuzovat podobnost transakcí.

7.2 Implementace modelu AI

Implementace modelu na serveru se inspiruje z [5]. Na serveru je modelu AI dedikován celý soubor `model.py`. Tento soubor nejdříve načte data a funkce `clean_text_bert` je následně očistí od zbytečných znaků a celou transakci promění na tokeny, které poté lépe dokáže zpracovat model. Poté následuje inicializace modelu z knihovny SentenceTransformer s názvem `paraphrase-multilingual-mpnet-base-v2`. Dále se vytvoří vektorové reprezentace textu pro vstupní data. Výsledkem je objekt `embeddings`, který obsahuje vektorové reprezentace textu.

```
df = pd.read_csv('dataset.csv')
text_raw = df['Transaction']
text_BERT = text_raw.apply(lambda x: clean_text_BERT(x))
bert_input = text_BERT.tolist()
model = SentenceTransformer('paraphrase-multilingual-mpnet-base-v2')
embeddings = model.encode(bert_input, show_progress_bar=True)
embedding_BERT = np.array(embeddings)
```

Výpis 7.1: `model.py`.

[2] https://pandas.pydata.org/
[3] https://www.nltk.org/

Při tvoření objektu transakce se volá funkce `ai_categorization(text)`. Funkce spustí model umělé inteligence a ten pomocí kosinové podobnosti a jazykového modelu BERT se pokusí najít dle kontextu slov podobnou transakci z datasetu a pomocí této transakce se určí kategorie pro transakci. Zkrácenou verzi funkce můžeme vidět na výpisu 7.2.

```
def ai_categorization(transaction_text):
    data = [transaction_text]
    df_test = pd.DataFrame(data, columns=['Test'])
    text_test_raw = df_test['Test']
    text_test_BERT = text_test_raw.apply(lambda x: clean_text_BERT(x))
    bert_input_test = text_test_BERT.tolist()
    embeddings_test = model.encode(bert_input_test, show_progress_bar=True)
    embedding_BERT_test = np.array(embeddings_test)
    df_embedding_bert_test = pd.DataFrame(embeddings_test)
    similarity_new_data = cosine_similarity(embedding_BERT_test,
        embedding_BERT)
    similarity_df = pd.DataFrame(similarity_new_data)
    data_inspect = df.iloc[index_similarity, :].reset_index(drop=True)
    category = data_inspect['Category']
    return category[0]
```

Výpis 7.2: Funkce `ai_categorization(text)`.

7.3 Trénování modelu

Pro správnou funkčnost kategorizace je potřeba natrénovat model pro umělou inteligenci na nějakém vhodném datasetu.
Vyextrahoval jsem své osobní transakční data z banky a také data od rodinných příslušníků a kamarádů. Takto vznikla kolekce dat, která obsahuje více než 5000 transakcí. Z transakcí jsem použil pouze sloupeček s popiskem transakce. Z mé datové analýzy jsem se dozvěděl, že supermarkety nemají stejná čísla bankovního účtu jako stejná značka supermarketu v jiném městě.
Rozhodl jsem se rozdělit transakce na tyto kategorie:

- **Potraviny**
- **Oblečení**
- **Pravidelné výdaje** – Energie, elektřina, voda, topení, splátky, hypotéky, nájem
- **Investice** – Investiční výdaje a odkládání peněz na spořící účet
- **Transport** – Jízdenky a MHD
- **Internetové nákupy** – Jakákoliv transakce, která prošla přes internetovou platební bránu
- **Palivo** – Benzín, nafta
- **Přeposílání peněz** – Převod peněz na další účet, nebo dobití peněz na Revolut
- **Bankomat** – Výběr peněz z bankomatu

- **Mobil** – Dobíjení kreditu, placení paušálu
- **Nákupy** – Všechno ostatní co nepatří do potravin a oblečení
- **Předplatné** – Noviny, časopisy, hudební a streamovací služby
- **Lékárna** – Léky
- **Zábava** – Lístky na představení, bar nebo výlety

Testovací data obsahují 100 transakcí z čehož se zde nachází, jak transakce známé v datasetu tak i neznámé. Známé transakce se zde nacházejí pro kontrolu správnosti.

V následujících kapitolách popíšu experimenty, které jsem provedl a který postup jsem nakonec zvolil do výsledné aplikace.

7.3.1 Trénovaní modelu na nevyčištěných datech

Neošetřená data obsahují mnoho záznamů, která jsou duplicitní, prázdná nebo nesmyslná pro potřeby trénovaní modelu např. obsahují směs znaků a číslic ze které nedokážeme vyvodit kategorii transakcí. Po této selekci transakcí jsem se dostal na číslo 400 použitelných transakcí. Transakce jsem ručně prošel a přidal k ním výše zmíněnou kategorii do které patří. Pokud jsem nedokázal posoudit kam transakce patří, zařadil jsem ji do kategorie Neznámé.

Po provedení experimentu jsem došel k závěru, že tato metoda nedává nijak obstojné výsledky a ze 100 transakcí bylo správně rozpoznáno 63 transakcí (z toho 40 bylo předem známých, 23 neznámých) a 37 transakcí bylo nepřesně zařazeno nebo zcela mimo obor (10 předem známých, 27 neznámých). Z trénovaní vyšlo najevo, že největší problém dělají umělé inteligenci jména a adresy, pokud je v popisku transakce pouze jméno, nedokáže najít přesnou shodu a pokud je tam jenom adresa, tak z toho taktéž nelze určit o co se jedná. Častá chyba umělé inteligence byla, že se snažila přiřadit kategorii dle místa. Konečná úspěšnost je 80 % na známých transakcích a 46 % na nových. Z těchto čísel se dá spočítat kombinovaná úspěšnost 63 %.

Příklady chybně kategorizovaných transakcí lze vidět v tabulce 7.1.

Nová transakce	Odpovídající transakce	Kategorie
ZHU TE MIAO	JIZDNE *UHBUkf1549	Transport
KB ATM UH.BROD KAUFL	Sklizeno	Potraviny
DECATHLON ZLIN	MCD Zlin	Jídlo
MOBELIX BRNO SPLAVIS	MOL - BRNO - 606	Palivo
HRACKY SOUKENIK	JIZDNE *UHBUmi1925	Transport

Tabulka 7.1: Příklady chybných transakcí v experimentu č.1

7.3.2 Trénovaní modelu na vyčištěných datech

V minulém experimentu jsem zaznamenal vysokou chybovost v datech, které nejsou očištěné např. „Tesco Brno“ a „McDonalds Brno“ (obě byly zařazeny do kategorie Potraviny). V tomto případě chybně rozezná kategorii a přiřadí ji podle města. Tento jev je chybný. Dalším problémem jsou čísla transakcí a různé další klasifikátory v popiscích transakcí

(např. „Revolut x4564d“). V tomto trénování modelu očistíme dataset od všech nepotřebných informací o transakci, jako jsou čísla objednávek a město, ve kterém se platba uskutečnila. Do datasetu jsem také přidal nejznámější obchodní řetězce působící v České republice a přiřadil k nim jejich kategorii. V tomto experimentu se špatně rozdělují transakce na již známé a na úplně nové, neboť pro ošetřené transakce neexistuje stejná transakce v testovacích datech, ale i přes tuto skutečnost můžeme porovnat celkovou úspěšnost. Po nacvičení modelu vyšla následující úspěšnost. Ze 100 testovacích transakcí bylo správně rozpoznáno 69 transakcí a 31 zcela špatně. Toto trénování dosahovalo mnohem lepších výsledků v oblasti adres, kde již nespojovala dvě transakce jenom podle lokace. Příklady chybně kategorizovaných transakcí lze vidět v tabulce 7.2. Z toho výsledku můžeme usoudit, že jsme dosáhli skoro 70% úspěšnosti. Proto je vhodné s tímto modelem pokračovat dále.

Nová transakce	**Odpovídající transakce**	**Kategorie**
ENFIN CZ A.S.	NOTINO	Nákupy
HANACKA HOSPODA	Zasilkovna	Nákupy
PRIMARK PRAGUE	Bageterie Boulevard	Jídlo
SHELL 8014	Revolut	Převod peněz
KAUFLAND CZ 2010	kfcrozvoz.cz	Jídlo

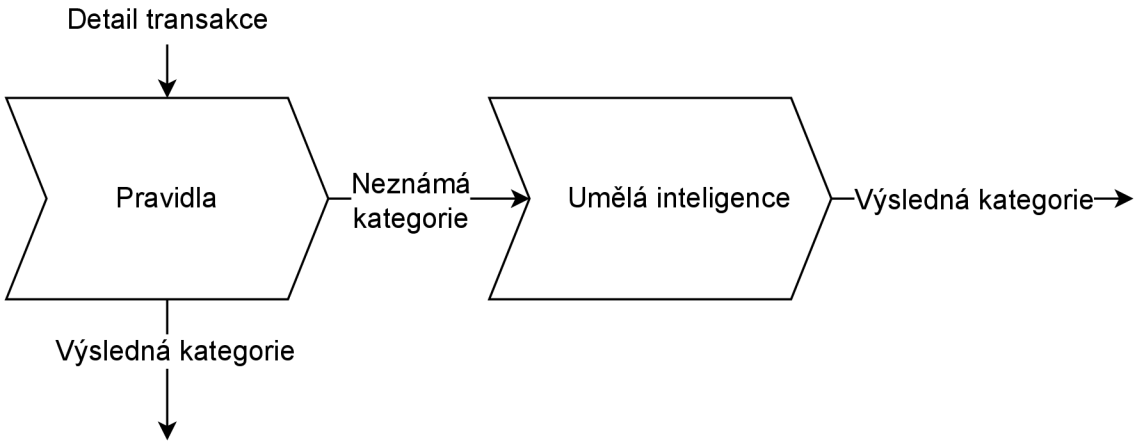
Tabulka 7.2: Příklady chybných transakcí v experimentu č.2

7.3.3 Kategorizace bez AI

V předchozích experimentech byl celý systém založen na umělé inteligenci. V této kapitole se však snažím přistoupit k této problematice jiným způsobem. Z velkého objemu transakcí vyberu ty nejčastěji se vyskytující a přidám k ní také seznam obchodním řetězců v České republice a vytvořím sadu pravidel. Z dat jsem dokázal vytvořit přes 200 pravidel, které obsahují ty nejčastěji používaná slova. Pokud část transakce bude obsahovat dané slovo, tak ho můžeme zařadit do dané kategorie. Zde nastává problém, pokud v transakci nenajdeme slovo, které by spustilo dané pravidlo, tak transakce spadne do kategorie Neznámé. Tento způsob se nedá dobře otestovat na daném testovacím vzorku, protože dopředu budeme vědět výsledek na základě daných pravidel. Tento způsob by se dal využít jako přídavná vrstva pro automatickou kategorizaci.

7.3.4 Kombinace AI a pravidel

V minulém experimentu jsem zmínil využití více vrstev pro kategorizaci transakcí. První detail transakce necháme projít pravidly, které jsem zmínil v minulé kapitole a poté použiji model AI z druhého experimentu a vzniknou nám dvě vrstvy, jak je ukázané na diagramu (obrázek 7.1). Z teoretického hlediska by tato kombinace měla přinést nejvyšší úspěšnost a eliminovat chyby umělé inteligence u předem známé transakce.



Obrázek 7.1: Vrstvená implementace kategorizace.

Z experimentu vyšlo najevo následující: Pokud použijeme testovací sadu, úspěšnost je velmi dobrá, neboť z původních transakcí se hodně obchodních společností opakuje a tak už byly zařazeny do pravidel. Můžeme říct, že pokud část transakce byla v pravidlech, je úspěšnost 100 %, pokud transakce nebyla kategorizována pravidly, je úspěšnost 70 %. Z toho vyplývá kombinovaná úspěšnost 85 % ve dvouvrstvé architektuře. Uvažujeme, že v běžném výpisu je průměrně 20-25 % transakcí, kterou nerozezná systém pravidel a pouze v těchto případech bude potřeba spouštět model umělé inteligence. Tyto data jsem dostal po datové analýze mých testovacích výpisů.

7.3.5 Závěr

Po provedení těchto tří experimentů jsem došel k závěru, že nejlepší je použit dvouvrstvou architekturu, protože předchozí experimenty ukázaly, že použití umělé inteligence ke kategorizaci transakcí nedávalo uspokojivé výsledky. V prvním experimentu měl model obtíže s identifikací správné kategorie, zejména pokud byl v popisu transakce uveden pouze název nebo adresa. V druhém experimentu, i když si model vedl lépe po vyčištění dat, stále docházelo k chybám při klasifikaci transakcí na základě místa. Z experimentů vyšlo najevo, že dvouvrstvá architektura nám zajistí nejlepší výsledky a zároveň server bude méně zatížen tím, že se nebude muset tak často provádět výpočet podobnosti v modelu umělé inteligence.

Při každém experimentu jsem si vedl poznámky k tomu, která kategorie byla nejčastěji špatně zařazena. Při prvním experimentu se nejčastěji netrefil u kategorie Nákupy. V této kategorii chyboval celkově 8krát. Ve druhém experimentu byly nejčastěji chybně zařazeny transakce kategorie Nákupy (8krát). Na rozdíl od prvního experimentu zde se zdvojnásobila chybovost u kategorie Potraviny. Můžeme konstatovat, že pokud model nevěděl, zkoušel neznámé položky zařadit mezi kategorie, které byly zastoupeny v datasetu častěji. Na základě těchto informací víme, že model se po úpravě datasetu choval víc předvídavě a nesnažil se výsledky více roztrousit jak tomu bylo u prvního experimentu. Toto chování by mohlo být považováno za významné zlepšení, neboť v případě, že v rámci transakce není správně rozpoznán význam určitého slova, existuje vysoká pravděpodobnost, že tato transakce spadá do kategorie nákupů či potravin, kde se vyskytuje mnoho malých prodejců.

Kapitola 8

Testování

Testování softwaru je důležitý proces při vývoji softwarových aplikací. Jeho účelem je ověřit, zda aplikace funguje tak, jak by měla, a zda splňuje požadované specifikace a funkční požadavky [24]. V následujících kapitolách rozeberu jakým způsobem byla aplikace testována.

8.1 Testování API

Testování probíhalo pomocí aplikace Postman[1]. Tato aplikace umožňuje vývojářům snadno otestovat API a získat informace o jeho funkcích a chování. Tato aplikace mně velmi pomohla ve zjištění, jak má API správně fungovat a reagovat na různé požadavky. Aplikace umožňuje si předpřipravit všechny endpointy a při každé změně v API jedním kliknutím vyzkoušet, zda jejich odpověď odpovídá změně chování, která byla provedena. Tímto způsobem probíhalo testování API po celou dobu vývoje.

8.2 Testování šablon

Aktuálně aplikace podporuje 5 českých bank (Komerční banka, ČSOB, Moneta, AirBank, Fio). Testovací výpisy pro tyto banky se mně podařilo získat od třech na sobě nezávislých osob pro každou banku. Provedl jsem analýzu výpisu a zaměřil se na to jak nejlépe vyextrahovat data z těchto šablon. Následně probíhalo testování na testovacích výpisech, výpisy ve všech třech vydání se shodovali ve své specifikaci a tak bylo testování úspěšné.

8.3 Uživatelské testování

Uživatelské testování probíhalo ve třech experimentech. Podmínkou pro účast v testování bylo vlastnit účet u jedné z podporovaných bankovních společností. Podařilo se mi najít 3 testovací subjekty, každý tvořil zástupce ze své věkové kategorie (18–30 let, 30–50 let, 50+ let). Testovací subjekty jsem označil jako Testovací subjekt 1 (18–30 let), Testovací subjekt 2 (30–50 let) a Testovací subjekt 3 (50+ let).

[1] https://www.postman.com/

8.3.1 Experiment č.1

V prvním experimentu měly testovací subjekty za úkol: Vytvořit si účet a nahrát výpis z minulého měsíce. U testovacích subjektů 1 a 2 jsem nenarazil na žádnou chybu a vše probíhalo hladce. Oba si přirozeným způsobem vytvořili účet a nahráli své výpisy, které si dopředu stáhli z internetového bankovnictví. U třetího testovacího subjektu byl mírný problém s jazykovou bariérou aplikace je celá v angličtině a tak starší člověk může mít problém s jazykem, po vysvětlení základních slovíček již bylo vše v pořádku a experiment pokračoval. Nakonec i třetí testovací subjekt zvládl dokončit úkol. Na základě výsledku experimentu lze konstatovat, že experiment lze považovat za úspěšný a základní úkol aplikace můžeme charakterizovat jako snadno zvládnutelný, jak potvrzují výsledky úspěšného absolvování úkolu všemi třemi testovacími subjekty.

8.3.2 Experiment č.2

Druhý experiment spočíval v upravení dat na profilu a úpravou měsíce v daném výpisu. V tomto experimentu splnil obě části úkolu bez nápovědy pouze testovací subjekt č.1. Následně subjekty 2 a 3 dokázaly bez problému upravit výpis. U subjektu 2 a 3 nastaly komplikace v hledání jak upravit profil uživatele. Po asi 2 minutách hledání jsem daným subjektům poradil, kde hledat. Tyto výsledky mě velmi překvapily a při návrhu jsem očekával, že bude stačit informace „Profile“ při najetí na ikonu myší. Na základě výsledků druhého experimentu můžeme tvrdit, že umístnění tlačítka pro úpravu profilu není vhodné pro starší ročníky. To by se dalo změnit např. velkým navigačním tlačítkem „Edit Profile“.

8.3.3 Závěr

Na základě výsledků provedených experimentů lze konstatovat, že základní principy aplikace jsou snadno zvládnutelné. Účastníci první testovací úkol zvládli bez problému, s výjimkou jazykové bariéry u staršího uživatele, které však byly úspěšně překonány. V rámci druhého experimentu bylo zjištěno, že umístění tlačítka pro úpravu profilu není vhodné pro starší uživatele. Tato skutečnost pravděpodobně nezpůsobí nepoužitelnost aplikace jako celku. Celkově lze tedy říci, že aplikace v uživatelském testování obstála a plní svůj účel.

Kapitola 9

Závěr

Cílem práce bylo navrhnout a vytvořit webovou aplikaci, která bude analyzovat transakce a řadit je do předem určených kategorií. Hlavní myšlenkou byla snaha pomoci uživatelovi se lépe orientovat ve svých financích. Významným úspěchem v kategorizaci transakcí je použití přístupu dvouvrstvé architektury, která dosahuje úspěšnosti 85 %. Před samotným začátkem vývoje proběhla hloubková analýza webových frameworků a také byly prozkoumány formáty výpisů konkrétních bankovních společnosti. Následovalo hledání vhodného způsobu jak automaticky kategorizovat transakce a testování různých přístupů k automatické klasifikaci. Před implementační částí práce jsem provedl návrh celé aplikace pomocí ER diagramu a Diagramu případů užití. Dále jsem vyvinul aplikaci, ve které si uživatel vytvoří účet a nahraje své bankovní výpisy. Tyto výpisy zařadí umělá inteligence do různých kategorií a uživatelovi se zobrazí graf, ve kterém může pozorovat své výdaje v různých kategorií. Uživatel může procházet již nahrané výpisy a porovnávat grafy z různých měsíců.

Práce mi dala nové poznatky ohledně webových frameworků na straně serveru i klienta a také jsem se naučil pracovat s umělou inteligencí v jazyce Python. Při vývoji byl kladen důraz na jednoduchost a uživatelskou přívětivost aplikace. Také jsem se snažil zúročit všechny zkušenosti, které jsem získal v mém studiu.

V této práci bych chtěl dále pokračovat, protože je zde mnoho věcí ke zlepšení. Nabízí se zde možnost aplikaci obohatit o co nejvíce bankovních společností jak z České republiky tak ze zahraničí. Toto vylepšení je snadné na implementaci (kapitola 6.1.2), ale časově náročné na získání informací o struktuře výpisů a získání testovacích výpisů. Další vylepšení by se nabízelo napojení na API bank, toto by vyžadovalo nastudovaní a zjištění, které banky nabízí API a zda by takový přístup byl možný v ČR, z důvodů smluvních podmínek bankovních společností. Dále se nabízí rozšíření o automatické značení opakovaných plateb a výplat. V budoucnosti má tato práce šanci se stát oblíbenou nekomerční aplikaci pro masovou společnost.
Aplikace je dostupná online na `moneytracker.petrhybl.com`.

Literatura

[1] *Stack overflow developer survey 2022* [online]. [cit. 2023-03-02]. Dostupné z: https://survey.stackoverflow.co/2022/#web-frameworks-and-technologies.

[2] Adobe Systems. *Document management — Portable document format — Part 2: PDF 2.0*. Standard ISO 32000-1:2008. International Organization for Standardization, December 2020 [cit. 2023-03-18]. Dostupné z: https://www.iso.org/standard/75839.html.

[3] Bartholomew, D. Mariadb vs. mysql. *Dostopano*. 2012, sv. 7, č. 10, s. 2014.

[4] Bray, T. *The JavaScript Object Notation (JSON) Data Interchange Format* [Internet Requests for Comments]. RFC 8259. RFC Editor, December 2017 [cit. 2023-03-15]. Dostupné z: https://www.rfc-editor.org/info/rfc8259.

[5] Cui, J. *Categorize Free-Text Bank Transaction Descriptions Using BERT*. Towards Data Science, leden 2023 [cit. 2023-04-15]. Dostupné z: https://towardsdatascience.com/categorize-free-text-bank-transaction-descriptions-using-bert-44c9cc87735b.

[6] Dasgupta, D., Roy, A., Nag, A. et al. *Advances in user authentication*. Springer, 2017. ISBN 3319588079.

[7] Devlin, J., Chang, M.-W., Lee, K. a Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv preprint arXiv:1810.04805*. 2018.

[8] Fedosejev, A. *React. js essentials*. Packt Publishing Ltd, 2015. ISBN 9781783551620.

[9] Garcia Martin, M. a Camarillo, G. *Extensible Markup Language (XML) Format Extension for Representing Copy Control Attributes in Resource Lists* [Internet Requests for Comments]. RFC 5364. RFC Editor, October 2008 [cit. 2023-03-18]. Dostupné z: https://www.rfc-editor.org/rfc/rfc5364.

[10] Gourley, D., Totty, B., Sayer, M., Aggarwal, A. a Reddy, S. *HTTP: the definitive guide*. O'Reilly Media, Inc., 2002. ISBN 9780613912594.

[11] H.Gomaa, W. a Fahmy, A. A. A Survey of Text Similarity Approaches. *International Journal of Computer Applications*. 2013, sv. 68, s. 13–18, [cit. 2023-04-14]. Dostupné z: https://www.academia.edu/9531934/A_Survey_of_Text_Similarity_Approaches.

[12] IBM. HTTP protocol. [online]. Říjen 2021, [cit. 2023-04-18]. Dostupné z: https://www.ibm.com/docs/en/cics-ts/5.3?topic=concepts-http-protocol.

[13] Jones, M. B., Bradley, J. a Sakimura, N. *JSON Web Token (JWT)* [RFC 7519]. RFC Editor, květen 2015. DOI: 10.17487/RFC7519. Dostupné z: https://www.rfc-editor.org/info/rfc7519.

[14] Koďousková, B. *Co je jednostránková webová aplikace (SPA) a kdy ji využít?* Leden 2020 [cit. 2022-11-28]. Dostupné z: https://www.rascasone.com/cs/blog/jednostrankova-webova-aplikace-spa.

[15] Koďousková, B. *Proč k vývoji webových aplikací použít technologii NodeJS?* Červenec 2020 [cit. 2023-04-13]. Dostupné z: https://www.rascasone.com/cs/blog/node-js-architektura-moduly-npm.

[16] Malý, M. *Node.js - s JavaScriptem na server - Zdroják.* Zdroják, září 2010 [cit. 2023-04-13]. Dostupné z: https://zdrojak.cz/clanky/node-js-s-javascriptem-na-server/.

[17] Massé, M. *Rest api design rulebook.* O'Reilly, 2012. ISBN 1449310508.

[18] Ravikiran, S. *A Complete Guide to ER Diagrams in DBMS.* Simplilearn, květen 2021 [cit. 2023-04-14]. Dostupné z: https://www.simplilearn.com/tutorials/sql-tutorial/er-diagram-in-dbms.

[19] Reimers, N. a Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, Listopad 2019. Dostupné z: https://arxiv.org/abs/1908.10084.

[20] Rescorla, E. *HTTP Over TLS* [Internet Requests for Comments]. RFC 2818. RFC Editor, May 2000 [cit. 2023-03-18]. Dostupné z: http://www.rfc-editor.org/rfc/rfc2818.txt.

[21] Rouse, M. *Three-Tier Architecture.* Leden 2021 [cit. 2023-04-14]. Dostupné z: https://www.techopedia.com/definition/24649/three-tier-architecture.

[22] Shafranovich, Y. *Common Format and MIME Type for Comma-Separated Values (CSV) Files* [Internet Requests for Comments]. RFC 4180. RFC Editor, October 2005 [cit. 2023-03-18]. Dostupné z: https://www.rfc-editor.org/rfc/rfc4180.

[23] Shwetank, D. *Web Storage: Easier, More Powerful Client-Side Data Storage.* Březen 2013 [cit. 2023-04-18]. Dostupné z: https://dev.opera.com/articles/web-storage/.

[24] Singh, S. K. a Singh, A. *Software testing.* Vandana Publications, 2012. ISBN 9788194111061.