Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Bankovní kalkulátor

Bc. Michal Černota

Diplomová práce
2023

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Michal Černota**
Osobní číslo:	**I20201**
Studijní program:	**N0613A140007 Informační technologie**
Téma práce:	**Bankovní kalkulátor**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Cílem diplomové práce je zpracování kalkulátoru bankovních poplatků pro český bankovní trh, a to dle výběru a specializace diplomanta buď pro běžné účty nebo pro studentské účty, které jsou nyní velmi žádaným produktem. Práce bude obsahovat srovnání jednotlivých účtů pomocí metod vícekriteriálního rozhodování, výstupem práce je online aplikace umožňující žadateli o bankovní účet srovnat nabídky jednotlivých bank a optimálně se rozhodnout na základě jeho subjektivních požadavků. Do metod rozhodování je možné zahrnout též rozhodování za neurčitosti, za rizika nebo jiné využití fuzzy množin.

Rozsah pracovní zprávy:
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná**

Seznam doporučené literatury:

REVENDA, Zbyněk. Peněžní ekonomie a bankovnictví. 5., aktualiz. vyd. Praha: Management Press, 2012, 423 s. ISBN 978-80-7261-240-6.
POLOUČEK, Stanislav. Bankovnictví. 2. vyd. V Praze: C.H. Beck, 2013, xvi, 480 s. Beckovy ekonomické učebnice. ISBN 978-80-7400-491-9.
KALABIS, Zbyněk. Základy bankovnictví: bankovnictví obchody, služby, operace a rizika. 1. vyd. Brno: BizBooks, 2012, 168 s. ISBN 978-80-265-0001-8. Online zdroje jednotlivých bank.

Vedoucí diplomové práce: **Mgr. Alena Pozdílková, Ph.D.**
Katedra matematiky a fyziky

Datum zadání diplomové práce: **8. listopadu 2021**
Termín odevzdání diplomové práce: **20. května 2022**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

prof. Ing. Antonín Kavička, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 30. listopadu 2021

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10. 5. 2023

Michal Černota

PODĚKOVÁNÍ

Chtěl bych poděkovat vedoucí této práce Mgr. Aleně Pozdílkové Ph.D. za její ochotu a rady v průběhu zpracování této diplomové práce. Dále bych rád poděkoval své rodině a příbuzným za jejich podporu během doby studia.

ANOTACE

Diplomová práce je zaměřena na analýzu bankovních účtů pomocí metod vícekriteriálního rozhodování a metod pro stanovení vah kritérií. Její teoretická část popisuje problematiku bankovnictví, vícekriteriálního rozhodování a hodnocení variant. V rámci praktické části práce byla implementována webová aplikace, která uživateli poskytuje možnost analýzy bankovních účtů, jejímž cílem je doporučení optimálního účtu na základě stanovených preferencí. V závěru jsou shrnuty výsledky analýzy bankovních účtů pro zvolené stavové modely.

KLÍČOVÁ SLOVA

Bankovní účet, vícekriteriální analýza variant, kritéria, varianty, webová aplikace

TITLE

Bank calculator

ANNOTATION

This diploma thesis focuses on the analysis of the bank accounts using multi-criteria decision making methods and methods for determining criteria weights. Theoretical part of the thesis describes theory of banking, multi-criteria decision-making and variant evaluation. In the practical part of the thesis, a web application has been implemented, which provides the ability to analyze bank accounts for a user, the aim of which is to recommend the optimal account based on the preferences set. The paper concludes with a summary of the results of the bank account analysis for the selected state models.

KEYWORDS

Bank account, multi-criteria variant analysis, criteria, variants, web application

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

AHP	Analytic Hierarchy Process
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
API	Application Programming Interface
REST	Representational State Transfer
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
DLL	Dynamic-link library
UI	User interface
JSON	Javascript Object Notation
HTTP	Hypertext Transfer Protocol
CRUD	Create-Read-Update-Delete
SOAP	Simple Object Access Protocol
HTML	Hypertext Markup Language
XML	Extensible Markup Language
OAuth	Open Authorization
JWT	Json Web Token
HMAC	Hash-Based Message Authentication Codes
RSA	Rivest-Shamir-Adleman
ECDSA	The Elliptic Curve Digital Signature Algorithm
ORM	Objektově relační mapování
SQL	Structured Query Language
LINQ	Language Integrated Query
SPA	Single-page application
WASM	WebAssembly
DOM	Document Object Model
DI	Dependency Injection
UML	Unified Modeling Language
SMTP	Simple Mail Transfer Protocol
ČR	Česká republika
EU	Evropská unie
SMS	Short Message Service
PDF	Portable Document Format

ÚVOD

Rozhodování je proces, se kterým se člověk setkává denně. Jedná se však většinou o nepříliš podstatná rozhodnutí, a proto se většina lidí rozmýšlí instinktivně a bez rozmyslu. Ať už je v těchto situacích učiněno jakékoli rozhodnutí, dá se očekávat, že volba nebude mít výrazný vliv na budoucnost jedince. V případě významných rozhodnutí, jako je například strategické rozhodnutí podniku, volba zaměstnání nebo například rozhodnutí o určité investici, je však vhodné zamyslet se nad následky zvolení konkrétní volby a případně použít určitý nástroj, který rozhodovateli pomůže vybrat takovou možnost, která je pro daného jedince nejvýhodnější. Neméně důležité je rozhodování v situacích týkajících se osobních financí. Většině lidí totiž na svých financích záleží, chtějí o ně správným způsobem pečovat, ochraňovat je a zacházet s nimi co nejlepším způsobem. Proto se jeví i zvolení optimálního bankovního účtu jako důležité rozhodnutí, zejména v současné době, která je nejen po ekonomické stránce velmi náročná.

Tato práce se zabývá vícekriteriálním rozhodováním a hodnocení variant a aplikací metod vícekriteriálního rozhodování na výběr bankovního účtu. V teoretické části bude popsána teorie bankovnictví se zaměřením na bankovní účty, jejich účel, využití a poplatky za využívání bankovních účtů. Teoretická část se kromě charakteristiky metod vícekriteriálního rozhodování a metod pro stanovení vah kritérií zaměřuje také na popis principu a způsobu jejich aplikace.

Cílem praktické části této diplomové práce je implementace webové aplikace, která bude uživateli umožňovat zvolení běžného či studentského bankovního účtu, který bude optimální vzhledem ke stanoveným preferencím uživatele. Dále bude poskytovat vhodný způsob zobrazení bankovních produktů, jejich poplatků a zobrazení dalších základních informací o bankovních účtech. V závěru bude provedena analýza vyhodnocení studentských i běžných bankovních účtů za pomoci zvolených metod vícekriteriálního rozhodování na předem určených stavových modelech.

1 BANKOVNICTVÍ

1.1 Historie peněz

Pro vysvětlení účelu bankovních účtů, v čem jsou výhodné a k rozlišení jejich typů, je vhodné nejprve zmínit funkci peněz. Peníze se staly součástí každodenního života jak v rovině individuální, tak makroekonomické a ovlivňují rozhodování každého ekonomického subjektu. Obecně lze říct, že peníze jsou cokoli, co slouží jako běžně přijímaný prostředek při placení. Peníze mají tři základní funkce – slouží jako prostředek směny, fungují jako zúčtovací jednotka a uchovávají hodnotu.

Peníze mají dlouhou historii, z čehož plyne, že během své existence prošly značným vývojem. Nejstarší podoba peněz byla ve formě komodit (drahé kovy, dobytek atd.), které byly směnovány mezi osobami, a postupně prošly rozvojem až do víceméně takové podoby, jak je známe dnes – čili do podoby kovových mincí a papírových bankovek. Výhoda kovových mincí a bankovek je ta, že na rozdíl od například směnného obchodu, jasně určují hodnotu komodit a jsou snadno přenositelné. Dále je potřeba zdůraznit, že mince i papírové oběživo jsou peníze s nuceným oběhem, a jsou tedy zákonným platidlem. Znamená to, že dle zákona musí být přijímány všemi subjekty jako prostředek úhrady. Nemohou navíc být vytvářeny soukromými subjekty, což způsobuje jejich vzácnost.

V dnešní době jsou s vývojem technologií čím dál více populární zejména takzvané bankovní peníze. Jde o bezhotovostní peníze, které existují na bankovních účtech a lze s nimi disponovat například prostřednictvím platebních karet. V dnešní době je přibližně devět z deseti transakcí provedena prostřednictvím bankovních peněz, kdežto pouze jedna připadá na hotovostní peníze. [3]

1.2 Banky

Banka je v zásadě podnikatelský subjekt se specifickými rysy, které ji odlišují od podniků z jiných odvětvích, což vyplývá zejména z jejich podstaty jakožto obchodníků se svěřenými penězi, a obvykle se k nim vztahují odlišná pravidla oproti obecné úpravě podnikání. Z tohoto důvodu je potřeba u daného ekonomického subjektu jednoznačně určit, zda se jedná o banku, či nikoli.

Banku lze charakterizovat jako druh finančního zprostředkovatele, jehož hlavní činnost je zprostředkování pohybu finančních prostředků mezi ekonomickými subjekty. Tento pohyb prostředků je založen na přijímání vkladů a poskytování úvěrů. Směrnice EU pak banku

definuje jako podnikatelský subjekt, jehož činností je přijímání vkladů nebo jiných peněžních prostředků od veřejnosti a poskytování úvěrů na vlastní účet.

Český zákon o bankovnictví stanovuje, že banky musí splňovat následující podmínky:

- právnická osoba se sídlem v České republice
- založena jako akciová společnost
- přijímá vklady od veřejnosti
    - vkladem se dle zákona (zákon č. 21/1992 Sb., o bankách, § 1, odst. 2) rozumí svěřené peněžní prostředky, představující závazek vůči jejich vkladateli na jejich výplatu
- poskytují úvěry
    - forma dočasně poskytnutých peněžních prostředků žadateli
- mají povolení působit jako banka
    - vedle obecných podmínek je zapotřebí získat povolení (bankovní licenci) opravňující provádění bankovní činnosti od České centrální banky

Zvláštním typem banky je centrální banka, jejíž hlavním cílem je péče o zdravý měnový vývoj. Centrální bankou v České republice je Česká národní banka, která vznikla 1. 1. 1993. Postavení centrální banky je v tuzemsku dáno Ústavou (Ústava České republiky, zákon č. 1/1993 Sb.), která zaručuje její nezávislost, a dále je konkretizováno v zákoně o České národní bance (zákon č. 6/1993 Sb., o České národní bance), který určuje její pravomoci, organizaci nebo například cíle. Mezi její konkrétní funkce patří emise hotovostního oběživa, provádění měnové politiky, bankovní regulace a dohledu, nebo například plní funkci banky státu. [3]

1.2.1 Bankovní produkty

Bankovní produkty jsou služby, které banky mohou nabízet a provádět zpravidla za úplatu. Tyto produkty jsou obvykle nemateriálního charakteru (tzn. nejsou viditelné a není je tedy možné skladovat či patentovat). Mohou však být mezi sebou propojeny (případně navzájem podmíněny). Podmíněnost jednotlivých produktů mohou banky využívat při stanovení cen za tyto produkty zejména tehdy, je-li vazba velmi těsná (při využívání jednoho produktu je víceméně automaticky vyžadováno používat jiný produkt). Cena některého z produktů pak může být velmi nízká (případně je služba zdarma) s cílem přilákat klienty.

Systematizace bankovních produktů není vzhledem k jejich vysokému počtu a rozsáhlé různorodosti příliš jednoznačná. Pravděpodobně nejrozšířenější členění bankovních produktů je dle jejich odrazu v bilanci banky. Dle tohoto kritéria se dělí na aktivní, pasivní a neutrální bankovní produkty. Aktivní bankovní produkty jsou ty, které se odrážejí v aktivech bilance banky. U těchto produktů vystupuje banka jako věřitel a vznikají jí pohledávky (např. při poskytnutí úvěru) nebo vlastnická práva. Pasivní bankovní produkty se naopak odrážejí v pasivech. Jedná se především o produkty, kdy banka získává na úvěrové bázi cizí kapitál, a klasickým pasivním bankovním obchodem je příjem vkladu. [3]

1.2.2 Cena bankovních produktů

Stejně jako v případě produktů z jiných odvětví, tak i v bankovnictví hraje cena produktu důležitou roli. Hlavním cílem cenové politiky banky je stanovit takové ceny produktů, aby znamenaly dostatečnou rentabilitu banky, zlepšily její postavení na bankovním trhu a odrážely nákladovost banky na tyto produkty.

Druhy cen bankovních produktů je vzhledem k nejednotnému označování obtížné definovat. Za základní členění lze ale považovat:

- úroky – cena za zapůjčení peněz bankou
- provize a prémie – cena za poskytnutí služby bankou, kde banka přebírá určité riziko
- přímé poplatky – cena za provedení služby, s kterou jsou pro banku spojené určité vyčíslené náklady (např. poplatky za vedení účtu)
- nepřímé poplatky – cena za poskytnutí služby, kde cena je skryta v jiné ceně

Cena za tyto produkty bývá většinou stanovena na určité bázi, ke které se produkt vztahuje, přičemž tyto báze lze také navzájem kombinovat. Výsledná cena se tak může odrážet na základě časové jednotky, po kterou je služba využívána, nebo dle hodnotového objemu. Konečná cena je poté určena součinem jednotkové ceny produktu a jeho hodnotového objemu. Cenu je však také možné určit paušálně pro jednotlivý dílčí produkt. V tom případě je dána součinem četností jednotlivých produktů a jednotkové ceny. Mezi produkty (respektive činnosti) naceněné tímto způsobem lze zařadit například poplatek za výběr z bankomatu nebo poplatek za provedení platebního příkazu. [3]

2 BANKOVNÍ ÚČTY

Bankovní účet je produkt, který odráží vztah mezi bankou a jejími klienty. Klientský bankovní účet lze charakterizovat jako účet pohledávek a závazků banky, vyplývajících ze vztahů mezi bankou a klientem. Pokud je banka v roli dlužníka (prostřednictvím bankovního účtu přijme peníze), pak prostřednictvím bankovního účtu eviduje svoje dluhy a závazky vůči konkrétnímu klientovi.

Existuje více druhů bankovních účtů, které vycházejí z účelu, ke kterému jsou využívány. Jejich členění nemusí být zcela identické. Jednotlivé banky mohou tyto účty různým způsobem modifikovat či kombinovat, například z důvodu zvýšení atraktivity těchto bankovních produktů pro své klienty. [3]

2.1 Druhy klientských bankovních účtů

Banky nabízí svým klientům různé účty, které se od sebe odlišují zejména účelem jejich použití. Rozdílnost se může skrývat také například v právních normách nebo v technikách jejich vedení. Banky navíc mohou tyto produkty různě kombinovat z důvodu zvýšení atraktivity nabízeného produktu pro své klienty. Pokud je veden v zahraniční měně, pak je nazýván devizovým, který se od účtu vedeného v tuzemské měně může lišit ve výši poplatků, právní úpravě, a výše úrokových sazeb odpovídá sazbám v dané měně.

Běžný účet lze považovat za základní bankovní produkt, jelikož je předpokladem pro využívání dalších produktů banky a vytvoření bankovního účtu lze pokládat za vznik vztahu mezi bankou a klientem. Uvedený produkt slouží především k uložení dočasně dostupných prostředků a provádění bezhotovostního platebního styku.

Kombinací běžného a úvěrového účtu vznikne tzv. kontokorentní účet, který jeho majiteli poskytuje stejné možnosti jako běžný účet. Jeho odlišnost však spočívá v možnosti čerpání kontokorentního úvěru, tj. čerpání hotovosti i v případě nedostatku prostředků na účtu. K evidenci úvěrů, které banka poskytla svým klientům, slouží úvěrové účty. Na depotních účtech vede banka pro své klienty cenné papíry, které u ní mají ve správě nebo v úschově. Spořící bankovní účet je produkt, který zhodnocuje volné finanční prostředky. [3]

2.2 Založení a vedení bankovního účtu

Bankovní účet si může založit prakticky kdokoli – fyzické i právnické osoby, podnikatelé i občané. Klientský bankovní účet může být veden v českých korunách nebo v cizí měně. Je-li

účet veden v cizí měně, nazývá se devizovým běžným účtem. Banky vedou běžné účty v těch měnách, ve kterých mají stanovené úrokové sazby.

Bankovní účet je zřízen na žádost klienta, a pro jeho vznik je zapotřebí písemná smlouva mezi bankou a jejím klientem, přičemž účet může být zřízen i pro několik osob. Jeli účet zřízen pro více osob, pak každá osoba se stává majitelem účtu. Při zřízení účtu je klientovi sděleno bankovní spojení neboli takzvané číslo účtu. Náležitostmi smlouvy o zřízení účtu jsou:

- den, ke kterému je účet zřízen – od tohoto data je klient oprávněn disponovat s prostředky na účtu
- měnu, ve které je účet veden – v této měně jsou také následně vypláceny úroky
- způsoby a podmínky disponování s prostředky
- podmínky uplatnění inkasního placení
- způsob a lhůty předkládání platebních příkazů
- výše a způsob úročení účtu
- forma, způsob a periodicita předávání zpráv o stavu a pohybech na účtu
- výše a způsob úhrady poplatků za vedení účtu
- podmínky vypovězení smlouvy
- další podmínky – například minimální počáteční vklad

Při uzavírání smlouvy o zřízení účtu je osoba zakládající účet povinna prokázat se platným průkazem totožnosti, za který je považován buď občanský průkaz, povolení k trvalému pobytu v České republice nebo cestovní pas. To platí i při založení anonymního účtu, u kterého je pouze omezen počet osob, které znají skutečnou totožnost klienta. Pokud je klientem, jenž zřizuje bankovní účet, právnická osoba, pak je navíc zapotřebí předložit doklad prokazující právní subjektivitu (výpis z trestního rejstříku), a v případě živnostníka je potřeba předložit živnostenský list. [3]

Informace o majiteli a jeho účtu podléhají bankovnímu tajemství a mohou být sděleny třetí osobě pouze na základě písemného souhlasu majitele. Bez souhlasu smí banka totožnost majitele a stav účtu sdělit pouze na vyžádání zákonem oprávněných orgánů. Mezi tyto případy patří například soudy pro účely občanského soudního řízení, notáři pro účely řízení o dědictví nebo celní úřady ve věci placení celních poplatků. [2]

Z uzavřené smlouvy následně vyplývají určité povinnosti ze strany banky vůči svému klientovi. Je například povinna přijímat na účet peněžité vklady a bezhotovostní platby v dohodnuté měně (není ale povinna provádět konverzi do jiné měny), provádět platby na základě příkazu majitele účtu, bez zbytečných odkladů opravit chybná zúčtování a podle stanovených lhůt smlouvy je povinna oznámit majiteli účtu údaje o přijatých a provedených platbách. Ze všeobecných podmínek vyplývá, že zprávy o zúčtování by měly obsahovat zejména označení klienta, datum provedení platby, identifikaci položek, počáteční a konečný zůstatek. [3]

2.3 Zrušení bankovního účtu

Běžný účet může být zrušen po uplynutí doby na kterou byl zřízen, splněním účelu, pro který byl sjednán, dohodou, výpovědí, a to buď ze strany klienta nebo banky, nebo okamžitým odstoupením banky od smlouvy. Klient může bankovní účet zrušit na pobočce banky, ale některé banky umožňují účet zrušit prostřednictvím internetového bankovnictví, nebo skrze infolinku.

Majitel účtu je oprávněn kdykoli písemně vypovědět smlouvu s okamžitou platnosti a bez udání důvodů. Jeho podpis na výpovědi však musí být úředně ověřen, nebyla-li předána osobně majitelem, jinak tento požadavek nebude banka akceptovat. Současně majitel předá bance, u které je účet zřízen, instrukce, jak má banka naložit se zůstatkem na účtu. Zůstatek na běžném účtu je vyplacen majiteli v hotovosti nebo převeden na jiný účet. Pokud majitel účtu do doby, kdy nabude platnost zrušení účtu účinnost, neoznámí, jak má být se zůstatkem na účtu naloženo, zůstatek je evidován, nikoli však úročen, a to až do doby promlčení dle § 361 obchodního zákoníku. Platby došlé po zrušení běžného účtu banka vrací zpět plátci.

Banka je také oprávněna zrušit bankovní účet klienta bez udání důvodů, například pokud není využíván v souladu se smlouvou. Banka smí vypovědět smlouvu s účinností nejdříve ke konci měsíce následujícího po měsíci, ve kterém byla výpověď doručena majiteli bankovního účtu. Lhůta, ve které bude účet zrušen, je stanovena bankou úměrně k dalším službám, které byly klientovi poskytovány v souvislosti s vedením tohoto účtu. Pro zrušení účtu je totiž nezbytné, aby byly s účtem také současně zrušeny tyto služby. [3]

2.4 Výnosy a náklady běžného účtu

Při rozhodování klienta o zřízení běžného účtu u konkrétní banky hrají nemalou roli výnosy a náklady, které klientovi účet přinese. V případě výnosů se jedná o takzvaný úrok. Do nákladů běžného účtu patří nejčastěji poplatky spojené s využíváním služeb účtu, jako například výběry z bankomatů, zaslání výpisu z účtu nebo poplatky za odchozí platby do zahraničí.

U běžných účtů platí, že ze zůstatku na účtu banka svým klientům vyplácí takzvaný úrok, přičemž výše úrokové sazby by měla být vymezena ve smlouvě o založení běžného účtu. Pokud smlouva neupravuje jinak, tak úroky jsou splatné koncem kalendářního čtvrtletí a jsou připisovány ve prospěch běžného účtu. Úroková sazba bývá z důvodu možnosti kdykoliv s vkladem disponovat velmi nízká. Z důvodu nízkého úročení je na účtech udržován zůstatek pouze ve výši potřebné pro platební styk. Úroky podléhají dani z příjmů, kde se u běžných účtů fyzických osob jedná o srážkovou daň 15 %. V případě běžných účtů právnických osob a fyzických osob – podnikatelů jsou úroky přičítávány do daňového základu a daní se společně s ostatními příjmy. Úročení prostředků na účtu vychází z časových okamžiků připsání, respektive odepsání prostředků z účtu, které se nazývají valutování. [3]

2.5 Platební karty

Platební karty lze charakterizovat jako moderní platební nástroj ve formě plastových karet, které odpovídají mezinárodním normám, a jejich držitel díky nim může provádět bezhotovostní platby či výběry hotovosti. Platební karta je pravděpodobně nejčastěji využívané rozšíření bankovního účtu, mezi jejíž hlavní výhody patří eliminace nedostatků hotovostního placení, snadnější a bezpečnější dispozice s prostředky na bankovním účtu. V České republice se lidé mohli s tímto platební nástrojem setkat již v roce 1968, kdy cestovní kancelář ČEDOK začala platební karty přijímat ve svých pobočkách. Větší popularity však toto rozšíření bankovního účtu získalo až po roce 2000.

Používání platebních karet přináší výhody pro jejich držitele, obchodníky i pro samotné banky. Z hlediska držitele lze mezi výhody bezesporu zařadit jednoduchost použití, zabezpečení a mezinárodní použití. Obchodníkům přináší především větší konkurenceschopnost a zvýšení obratu. Pro banku představují platební karty další zdroj výnosů (poplatky za vydání karty, poplatky za transakce, úrok při čerpání úvěru).

Ačkoli jsou využívány především spotřebitelským sektorem, tak je mohou využívat i podnikatelé či právnické osoby. Je potřeba také zmínit, že platební karty mohou kromě bankovních institucí vydávat také nebankovní instituce, kterou je například American Express.

Skutečnost, že banky dříve emitovaly vlastní platební karty, instalovaly vlastní bankomaty a vytvářely obchodní místa, přičemž tyto systémy nebyly kompatibilní se systémy jiných bank, býval problém. Tyto nedostatky však byly eliminovány vznikem bankovních sdružení usilujících o zavedení jednotných platebních karet. Z důvodu potřeby mezinárodního využití vznikají mezinárodní sdružení bank, kde banky v rámci tohoto systému vydávají jednotné

karty, které přizpůsobují svým obchodním podmínkám. Mezi tato sdružení patří například VISA nebo MasterCard.

Zhotovení platební karty je definováno mezinárodní normou ISO 3554, která určuje materiál, ze kterého je karta vyrobená, její rozměry a formu, ve které budou uvedeny potřebné informace. Údaje, které lze na bankovní kartě nalézt, jsou vydavatel platební karty, k jakému systému platebních karet karta patří, číslo karty, období platnosti, podpisový vzor a jméno držitele. Na zadní straně karty se nachází magnetický proužek, který slouží jako záznamové médium potřebných údajů pro elektronické transakce. Obsahuje takzvané servisní kódy, které umožňují technickým zařízením rozpoznat, zda je přípustné použití karty ještě před provedením autorizace transakce. Mimo jiné je na něm uloženo také personální identifikační číslo, známé pod zkratkou PIN. [3]

2.5.1 Typy platebních karet

Platební karty lze rozlišovat podle několika jejich vlastností, jako například dle vydavatele karty, způsobu zúčtování transakcí nebo podle druhu záznamu na kartě. Neznámější dělení je pravděpodobně dle zúčtování transakcí, které se dělí na charge karty, úvěrové a debetní. Debetní karta je v České republice nejčastějším typem karty, jenž umožňuje čerpat peníze z bankovního účtu, přičemž poskytuje okamžitou kontrolu, zda klient nepřekračuje kreditní zůstatek na účtu. Lze tak disponovat pouze dostupnými prostředky klienta. Kreditní karta umožňuje čerpat peníze do mínusu a banka tak poskytuje klientovi úvěr. Pro tyto půjčky většinou platí bezúročné období, které bývá delší než jeden měsíc. Charge karta je nejstarším typem platební karty a k placení dochází na základě čerpání měsíčního limitu dle měsíčního výpisu, a to bankovním převodem. [1]

3 VÍCEKRITERIÁLNÍ ROZHODOVÁNÍ A HODNOCENÍ VARIANT

Rozhodování je proces, se kterým se jedinec potká na denní bázi. Se situacemi, kdy je potřeba z několika variant vybrat optimální, se můžeme setkat v zaměstnání nebo například v domácnosti. V těchto případech se ale většinou jedná o rozhodování, kdy rozhodovatel není vystaven velkému riziku, a může tak jednat bez rozvahy, dle intuice. Existují však situace, kdy volba nevhodné varianty může vést ke komplikacím, jako je například snížení profitu podniku, úbytku popularity ve společnosti nebo ke zdravotním komplikacím. V těchto situacích, kdy se při výběru varianty rozhodovatel vystavuje určitému riziku, je potřeba se rozhodnout racionálně, a vybrat vhodnou variantu takovým způsobem, aby byla minimalizována pravděpodobnost nesprávného rozhodnutí.

Vícekriteriální rozhodovací problémy jsou popsány množinou n variant $X = \{x_1, x_2, \ldots x_n\}$, které jsou posuzovány na základě m kritérií z množiny $K = \{K_1, K_2, \ldots K_m\}$. Z těchto informací lze formulovat vícekriteriální model rozhodování, který je dle charakteristiky množiny variant buď diskrétní, kdy jsou varianty ohodnoceny dle kritérií, nebo spojitý, který má množinu variant vyjádřenou soustavou omezujících podmínek, do kterého je potřeba zahrnout i dodatečnou informaci o subjektivních preferencích rozhodovatele. Z toho vyplývá, že ke kritériím se vztahují i další důležité informace, jako například jejich váha, význam nebo vztah k ostatním kritériím.

Varianta, která by dosáhla současně nejlepšího hodnocení ve všech kritérií, se označuje jako ideální. Tato varianta ale většinou v množině variant neexistuje, a je zapotřebí vyhledat alternativní nedominovanou variantu, ke které v množině variant neexistuje jiná varianta, která je lépe hodnocena alespoň podle jednoho kritéria, a ne hůře podle ostatních kritérií. K výběru těchto variant je většinou zapotřebí preferenční informace rozhodovatele. Opakem ideální varianty je takzvaná bazální varianta. Jedná se o variantu, která má nejnižší ohodnocení podle každého kritéria. [4] [5]

3.1 Sestavení souboru kritérií

Tvorba souboru kritérií může být složitý proces a je potřeba dodržet několik zásad. Kritéria jsou takové charakteristiky variant, které skutečně představují hlediska hodnocení variant k danému cíli. Do konečného souboru kritérií mohou být zahrnuta i taková kritéria, která nejsou snadno měřitelná a pro získání jejich hodnot je potřeba znalost experta, jsou-li nepostradatelné pro ohodnocení variant. Dále je potřeba, aby soubor kritérií neobsahoval nadbytečná kritéria, která

lze odvodit z ostatních kritérií. Pokud je to možné, je vhodné se vyhnout závislostem mezi kritérii. Požadavek na úplný soubor kritérií může být konfliktní s požadavkem na minimální počet kritérií, který závisí na konkrétním problému. Menší počet kritérií však vede k větší průhlednosti modelu hodnocení.

Důležitou vlastností vybraného kritéria je jeho měřitelnost. Vybrané kritérium musí být porovnatelné s ostatními kritérii variant ať už na kardinální nebo ordinální stupnici, a je potřeba, aby bylo možné vyjádřit hodnocení variant k těmto kritériím. Hodnoty těchto kritérií navíc musí mít jasně definovaný obsah (například uvedení jednotky).

Metoda vhodná pro vytvoření kvalitního souboru kritérií, který splňuje požadované zásady, je vytvoření takzvaného stromu kritérií. Tomuto stromu jsou nejdříve přiřazeny značně abstraktní kritéria, která jsou postupně vyjádřena pomocí množin konkrétnějších kritérií. Tento proces se opakuje do doby, dokud nejsou nalezena atomická, přímo měřitelná kritéria, která jsou posléze použita pro hodnocení variant. Výsledkem je graf typu strom, kde uzly nacházející se ve vyšší hloubce stromu jsou více abstraktní a listy představují měřitelná kritéria. [4]

3.2 Metody stanovení vah kritérií

Kritéria jsou charakteristiky variant, na jejichž základě lze varianty navzájem posuzovat vzhledem k danému cíli hodnocení. Dělí se na kvalitativní, která jsou většinou zadávána slovně a popisují rozdílnou kvalitu vlastnosti, a kvantitativní, která jsou zadávána číselně a určují tak kvantitu vlastnosti.

Preference kritérií mohou být vyjádřeny několika způsoby, a to pomocí:

- aspiračních úrovní kritérií (nominální informace),
- pořadí kritérií (ordinální informace),
- vah kritérií (kardinální informace),
- nebo nemusí být stanoveny vůbec.

Váhy kritérií jsou nezáporná reálná čísla, která vyjadřují rozdílnou významnost vybraných kritérií vzhledem k cílovému hodnocení variant. Většinou se pracuje s normovanými váhami, pro které platí:

$$\sum_{j=1}^{m} v_j = 1$$

V problematice vícekriteriálního rozhodování se lze také setkat s případy, kdy informace o preferencích v množině kritérií zcela chybí, nebo je dána prostřednictvím preferenční relace na množině kritérií. [4], [6]

3.2.1 Fullerova metoda

Tato metoda stanovení vah kritérií využívá párového srovnávání kritérií, kde počet srovnání v případě n kritérií je roven počtu $\binom{n}{2}$, a její princip je založen na porovnávání vždy dvou kritérií, přičemž z této dvojice je vybráno významnější kritérium, a váhy jsou tak odvozeny z preferenční relace na množině kritérií. Je nazývána Fullerovou metodou proto, že při aplikaci této metody se využívá takzvaný Fullerův trojúhelník – jedná se o matici preferencí P, která znázorňuje významnější kritérium z každé možné dvojice, přičemž je definován pouze horní trojúhelník matice a zbytek matice lze následně odvodit. Pro prvek $p_{j,k}$ této matice platí, že

$$p_{j,k} = \begin{cases} 1 \text{ pokud je j-té kritérium významnější než k-té,} \\ 0 \text{ v opačném případě.} \end{cases}$$

Nenormovaná váha j-tého kritéria w_j, která určuje jeho významnost, je odvozena z počtu kritérií, před nimiž je toto kritérium upřednostněno, a vypočítat tuto váhu lze dle vzorce

$$w_j = \sum_{k=1}^{n} p_{j,k} + 1,$$

kde přičtená jednička zabraňuje tomu, aby nejméně významné kritérium mělo nulovou váhu.[4]

3.2.2 Saatyho metoda

Saatyho metoda se liší od metody párového srovnávání v tom, že místo matice preferencí P je zadávána matice intenzit S, jejíž prvky představují relativní významnost j-tého kritéria ke k-tému kritériu. Při zadávání hodnot této matice je využita obvykle pětibodová stupnice intenzit preferencí

$$s_{j,k} = \begin{cases} 1 \text{ pokud mají kritéria stejnou významnost,} \\ 3 \text{ slabá preference j-tého kritéria před k-tým,} \\ 5 \text{ silná preference j-tého kritéria před k-tým,} \\ 7 \text{ velmi silná preference j-tého kritéria před k-tým,} \\ 9 \text{ absolutní preference j-tého kritéria před k-tým.} \end{cases}$$

Pokud je *j*-té kritérium méně významné než *k*-té, pak hodnota prvku $s_{j,k}$ odpovídá hodnotě

$$s_{j,k} = \frac{1}{s_{k,j}}.$$

Hodnoty 2, 4, 6 a 8 lze použít pro hodnocení mezistupňů, je-li to vyžadováno. [4]

3.2.3 Metoda pořadí

Jedná se o jednoduchou metodu stanovení vah kritérií, která vyžaduje pouze ordinální informaci neboli určení pořadí kritérií dle jejich významnosti. Uspořádaná kritéria jsou ohodnocena body dle jejich pořadí, kde nejdůležitější kritérium má nejvyšší hodnotu a nejméně významné naopak hodnotu nejnižší. Při počtu kritérií *n* bude mít nejvýznamnější kritérium právě hodnotu rovnou *n*. Nejméně podstatné kritérium má ohodnocení rovno jedné. Váha kritéria se vypočte dle vzorce [5]

$$v_i = \frac{b_i}{\sum_{i=1}^{k} b_i}.$$

3.2.4 Bodovací metoda

Pro použití bodovací metody je vyžadován předpoklad kvantitativního ohodnocení kritérií zadavatelem dle jejich důležitosti. K tomuto ohodnocení je zvolena bodová stupnice, přičemž jsou kritéria ohodnocena dle jejich důležitosti hodnotami, které spadají do zvoleného intervalu. K ohodnocení mohou být využita reálná čísla a několika kritériím lze přiřadit stejnou hodnotu. Výhodou této metody oproti metody pořadí je ta, že umožnuje podrobněji vyjádřit preference zadavatele. Váhy kritérií se vypočítají stejným způsobem, jako v případě metody pořadí [5]

$$v_i = \frac{b_i}{\sum_{i=1}^{k} b_i}.$$

3.3 Diskrétní modely rozhodování

Diskrétní rozhodovací modely jsou explicitně popsány množinou variant $A = \{a_1, a_2, \ldots, a_p\}$, seznamem kritérií $F = \{f_1, f_2, \ldots, f_k\}$ a ohodnocením variant dle jednotlivých kritérií. Hodnocení variant dle jednotlivých kritérií lze vyjádřit dle takzvané kriteriální matice

$$Y = \begin{array}{c} \\ a_1 \\ a_2 \\ \ldots \\ a_p \end{array} \begin{array}{c} \begin{array}{cccc} y_1 & y_2 & \cdots & y_k \end{array} \\ \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1k} \\ y_{21} & y_{22} & \cdots & y_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ y_{p1} & y_{p2} & \cdots & y_{pk} \end{bmatrix} \end{array},$$

jejíž prvky y_{ij}, $i = 1, 2, \ldots p$, $j = 1, 2, \ldots, k$ vyjadřují informaci o ohodnocení variant podle definovaných kritérií. Tyto informace mohou být však vyjádřeny několika způsoby. Kardinální informace je vyjádřena skutečnou hodnotou, které varianty dosáhly při ohodnocení dle kritérií. Ordinální informace určuje pořadí varianty určené na základě kritérií. Relativní informace párově porovnává varianty mezi sebou dle vybraných kritérií. Cílem je najít takovou variantu, která dosahuje nejlepšího hodnocení podle všech kritérií. [5]

3.3.1 Rozhodování při jistotě

Rozhodování při jistotě znamená, že výběr určité varianty s sebou přináší určitý důsledek. V případě, kdy je explicitně známá kriteriální funkce, lze optimální variantu nalézt párovým porovnáním variant pomocí této funkce. To znamená nalézt optimální variantu, pro kterou platí

$$f(a_j) \geq f(a_i), pro\ i = 1, 2, \ldots, p.$$

V případě, kdy tato kriteriální funkce není známá, je potřeba vycházet z párových srovnávání variant a sestavit funkci užitku. Výběr varianty přináší pro rozhodovatele určitý užitek, který se měří dle funkce užitku $u(a)$, kde funkční hodnoty leží v intervalu $<0,1>$. Čím je varianta výhodnější, tím je výsledná hodnota funkce užitku varianty vyšší.

Ordinální funkce užitku je funkce, pro níž platí

$$u(a) > u(b) <=> a\ P\ b,$$
$$u(a) = u(b) <=> a\ I\ b.$$

Pokud je tedy varianta a ostře preferovaná před variantou b, pak hodnota funkce užitku pro variantu a je ostře větší než pro variantu b. V případě indiferentních variant jsou hodnoty funkce užitku pro obě varianty shodné.

V případě, kdy pro libovolnou čtveřici variant platí vztah

$$u(a) - u(b) > u(c) - u(d),$$

tedy varianta a je preferována před variantou b více než varianta c před d, tak se jedná o kardinální funkci užitku. Kardinální funkce užitku tedy vyjadřuje nejen preference mezi variantami, ale také jejich intenzitu.

Jedním ze základních principů v teorii rozhodování je princip maximalizace užitku, který rozhodovateli doporučuje vybrat právě tu variantu, která maximalizuje hodnotu funkce užitku na množině variant. [5]

3.3.2 Rozhodování při neurčitosti

Existují případy, kdy důsledek rozhodnutí není znám s jistotou. Může totiž záviset na stavu systému, přičemž existence těchto stavů je náhodným procesem. Rozhodování při neurčitosti vychází z existence n náhodných možných stavů $S_1, S_2, \ldots, S_n$,, jejichž pravděpodobnost výskytu není známa. Tuto rozhodovací situaci lze vyjádřit pomocí matice, jejíž řádky odpovídají variantám $a_1, a_2, \ldots, a_n$, sloupce možným stavům $s_1, s_2, \ldots, s_n$, a prvky matice $a_{ij}, i = 1, 2, \ldots, p, j = 1, 2, \ldots, n$, představují ohodnocení důsledků rozhodnutí za předpokladu, že nastala situace s_j.

V případě neexistence informace o pravděpodobnostním rozdělení výskytů stavů lze vycházet z principu ekvivalentní pravděpodobnosti (takzvané Laplaceovo kritérium). Tedy že pravděpodobnost výskytu každého stavu je identická:

$$p_j = 1/n, j = 1, \ldots, n.$$

Poté je vybrána varianta, která maximalizuje střední hodnotu plynoucí z výběru i-té varianty.

Optimistické pravidlo vycházející z předpokladu, že nastane nejpříznivější stav, je založeno na optimismu rozhodovatele a nazývá se princip maximaxu. Výběr varianty je tedy určen dle nejvyšší hodnoty v matici variant. Princip maximinu je naopak založen na pesimismu rozhodovatele a předpokládá, že nastane nejméně příznivý stav. V tomto případě je výběr varianty určen výběrem minimálních prvků v řádcích matice variant a následně je z těchto minimálních hodnot vybrána nejvyšší hodnota. Princip ukazatele optimismu (neboli Hurwitzovo pravidlo) je kombinací principu maximinu a maximaxu a využívá koeficient optimismu, který měří míru optimismu rozhodovatele při výběru variant. Princip minimaxu ztráty (Savageovo kritérium) vychází z principu ztracené příležitosti, kde ztráta vzniká v případě výběru nevhodné volby pro určitý stav. Tento princip je založen na matici ztrát, kterou lze sestavit dle vzorce

$$z_{ij} = \max_k a_{kj} - a_{ij}$$

tak, že od nejvyšší hodnoty prvku v každém sloupci jsou odečteny hodnoty prvků matice. Pro každou variantu je následně vybrána největší ztráta a poté je zvolena varianta s nejmenší ztrátou. [5]

3.3.3 Rozhodování při riziku

Stejně jako v případě rozhodování při neurčitosti, tak i rozhodování při riziku vychází z existence *n* možných náhodných stavů. V tomto případě je však známé pravděpodobnostní rozdělení těchto stavů. Tuto situaci můžeme opět modelovat pomocí matice, jejíž řádky představují varianty $a_1, a_2, \ldots, a_n$, sloupce odpovídají možným stavům $s_1, s_2, \ldots, s_n$, a prvky této matice a_{ij} představují ohodnocení důsledku výběru varianty za předpokladu, že nastala určitá situace.

Při rozhodování při riziku je využíván princip maximalizace očekáváné hodnoty, který je zobecněním principu ekvivalentních pravděpodobností v případě známého rozdělení pravděpodobností. Dle tohoto principu je vybrána varianta, maximalizující ukazatel

$$EV = \max_i \sum_{j=1}^{n} a_{ij} p_j.$$

Tento typ rozhodování bývá v některých případech doplněn o Bayesovskou analýzu, která na základě zkonstruované matice ztrát (podobně jako u principu minimaxu ztráty v případě rozhodování za nejistoty) určuje očekávanou ztrátu za předpokladu znalosti rozdělení pravděpodobnosti stavů. [5]

3.4 Spojité modely rozhodování

V předchozích rozhodovacích modelech byl předpoklad existence diskrétní množiny variant. Existují však případy, kdy je varianta popsána vektorem parametrů, které se mohou spojitě měnit. Varianta je tedy zadána vektorem parametrů

$$x = (x_1, x_2, \ldots, x_n)$$

a množina variant je vyjádřena soustavou omezujících podmínek, kde každá varianta, která splňuje soustavu omezujících podmínek, je variantou rozhodovací a tato varianta se tak stává přípustným řešením úlohy. Dále je předpokládána existence kriteriální funkce, která přiřazuje vektoru proměnných hodnoty, pomocí nichž je varianta ohodnocena. Cílem je pak najít takovou variantu, která je nejlépe ohodnocena, tedy nalézt optimální řešení úlohy.

Jedním typem spojitého modelu rozhodování je takzvaný model lineárního programování. Cílem úlohy je nalézt maximum kriteriální funkce

$$z = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$$

za stanovených omezujících podmínek

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &\leq b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &\leq b_m \end{aligned}$$

a za stanovení podmínky nezápornosti

$$x_j \geq 0\,, j = 1, 2, \ldots, n.$$

Jak bylo zmíněno, přípustné varianty jsou všechny, které vyhovují omezujícím podmínkám a podmínce nezápornosti.

Úlohy lineárního programování lze také řešit pomocí simplexové metody. Jedná se o iterativní metodu neboli optimální řešení je nalezeno postupně. Její algoritmus se skládá ze tří etap, a sice nalezení základního řešení, rozhodnutí, zda je řešení optimální, a na základě tohoto rozhodnutí je zapotřebí provést změnu výpočtu nebo výpočet ukončit v případě, že byla nalezena optimální varianta. [5]

3.5 Metody vícekriteriálního rozhodování

3.5.1 Metody nevyžadující informaci o preferenci kritérií

Ve chvíli, kdy jsou známé preference mezi variantami na základě kritérií, avšak nejsou známé preference mezi kritérii samotnými, pak lze použít metodu bodovací nebo metodu pořadí, které jsou velmi jednoduché a jejich aplikace je velmi podobná jako v případě stanovení vah kritérií pomocí těchto metod.

Všechny varianty jsou ohodnoceny dle každého kritéria číslem b_{ij}. V případě metody pořadí jsou varianty ohodnoceny číslem jedna až n, kde n udává počet variant. Bodovací metoda kvantifikuje informace na základě předem stanovené bodovací stupnice. Výsledné ohodnocení varianty je součtem hodnocení kritérií, tedy dle vzorce

$$b_i = \sum_{j=1}^{k} b_{ij}.$$

Varianty jsou následně seřazeny dle jejich ohodnocení. Varianta s nejnižším pořadovým číslem je nejvíce preferovanou variantou. [6]

3.5.2 Metody vyžadující aspirační úrovně kritérií

Metody spadající do této kategorie jsou použitelné, známe-li nominální informace o kritériích a kardinální ohodnocení variant podle kritérií.

Tyto metody jsou založeny na porovnávání kriteriálních hodnot variant s aspiračními úrovněmi kritérií. Ve většině případů jsou varianty rozděleny do dvou skupin: do skupiny neakceptovatelných variant, jejichž kriteriální hodnoty jsou horší než nastavené meze, a do skupiny akceptovatelných variant, která má naopak hodnoty kritérií lepší než tyto meze. V případě vhodně zvolených mezí může nastat situace, kdy v množině akceptovatelných variant zůstane jediná kompromisní varianta. Může však nastat i opačná situace, kdy není žádná varianta akceptována, a je tak potřeba uvolnit některé hodnoty aspiračních úrovní.

Konjunktivní a disjunktivní metoda jsou velmi podobné metody pro výběr akceptovatelných variant. Liší se však ve způsobu určení, zda varianta je akceptovatelná dle aspiračních úrovní. Konjunktivní metoda připustí pouze ty varianty, které splňují všechny aspirační úrovně, kdežto disjunktivní metoda připustí ty varianty, které splňují alespoň jeden požadavek.

Dále lze využít takzvanou metodu bazické varianty. Pro její aplikaci je potřeba určit váhy kritérií a vytvořit užitkovou funkci, která spočívá v porovnávání hodnot kritérií vzhledem k hodnotám bazické varianty. Užitek kritéria s klesající preferencí i-té varianty lze spočítat dle vzorce

$$u_{ij} = \frac{y_{ij}}{y_j^B},$$

kde hodnota j-tého kritéria bazické varianty je označena jako y^B_j a hodnota y_{ij} udává hodnotu j-tého kritéria i-té varianty. Užitek kritéria s rostoucí preferencí lze spočítat obdobným způsobem:

$$u_{ij} = \frac{y_{ij}}{y_j^B}.$$

Pro každou variantu je posléze spočten agregovaný užitek

$$u(y) = \sum_{j=1}^{m} v_j u_j(y_j),$$

na jehož základě je určeno konečné pořadí variant. [6]

3.5.3 Metody vyžadující ordinální informace

Některé metody pracují s ordinální informací o kritériích nebo variantách, vyžadují určení pořadí důležitosti kritérií a také určení pořadí variant podle těchto kritérií. Nejvyužívanější metoda této kategorie je lexikografická metoda, která je velmi jednoduchá pro aplikaci a vychází z předpokladu, že největší vliv na výběr kompromisního kritéria má nejdůležitější kritérium. Nastane-li případ, že je více variant ohodnoceno dle nejdůležitějšího kritéria stejně, pak se aplikuje stejný postup na druhé nejdůležitější kritérium. Postup se opakuje do té doby, dokud není vybrána jediná varianta, nebo dokud se tímto způsobem nevyčerpají všechna kritéria. V případě vyčerpání všech kritérií jsou jako kompromisní varianty označeny ty varianty, které zůstaly stejně ohodnoceny. [6]

3.5.4 Metody vyžadující kardinální informaci

Mezi základní metody, které vyžadují zadání kardinální informace o kritériích (neboli jejich vah) a informaci o variantách ve formě kriteriální matice s kardinálními hodnotami, patří přístupy hodnocení variant podle maximalizace užitku, vzdálenosti od ideální varianty a preferenční relace.

Metoda maximalizace užitku předpokládá, že zvolení určité varianty je možné vyčíslit jako užitek, který by přinesla realizace této varianty, a to reálným číslem v intervalu *<0; 1>*. Aby toto bylo možné, je nejprve zapotřebí stanovit pro každé kritérium takzvanou dílčí funkci užitku. Kardinální hodnocení variant podle kritérií bude tím pádem nahrazeno hodnotami dílčí funkce užitku

$$u_{ij} = u_j(y_{ij}), j = 1, 2, \ldots, n,$$

kde $u_j(y_{ij})$ je funkční závislost mezi původními kardinálními hodnotami matice kritérií a hodnotami dílčí funkce užitku a hodnota u_j je hodnotou dílčí funkce užitku. Pokud je hodnota funkce užitku pro kritérium rovna jedné, pak kritérium disponuje ideální hodnotou. Naopak pokud je hodnota funkce užitku rovna nule, tak se jedná o bazální hodnotu. Základními typy funkcí užitku jsou lineární, progresivní a degresivní funkce.

Metoda váženého součtu je konkrétním případem metody, která vyhodnocuje varianty na základě principu maximalizace užitku, a jak bylo zmíněno, také vyžaduje kardinální informaci, kriteriální matici a vektor vah. To znamená, že dosáhne-li varianta a_i určité hodnoty y_{ij}, tak pro uživatele přináší určitý užitek. Celkový užitek varianty je roven součtu hodnot dílčích funkcí užitku

$$u(\mathrm{a_i}) = \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{v_j u_j(y_{ij})},$$

kde hodnota u_j je dílčí funkce užitku kritérií a v_j jsou váhy těchto kritérií.

Pro aplikaci této metody je nejprve zapotřebí určit ideální variantu *H* a bazální variantu *D* s příslušnými hodnotami kritérií. Dále je vytvořena standardizovaná kriteriální matice *R* podle vzorce

$$r_{ij} = \frac{y_{ij} - d_j}{h_j - d_j}.$$

Prvky matice *R* představují hodnoty funkce užitku i-té varianty podle j-tého kritéria. Nakonec je pro každou variantu vypočtena agregovaná funkce užitku

$$u(a_i) = \sum_{j=1}^{n} v_j r_{ij},$$

a varianty jsou seřazeny podle hodnot $u(a_i)$. Varianta s nejvyšší agregovanou hodnotou užitku je konečným řešením.

Metoda AHP (Analytický hierarchický proces) je vhodná pro řešení komplexních rozhodovacích problémů rozkladem složité nestrukturované situace na jednodušší komponenty, čímž vytváří hierarchickou strukturu problému. Na každé úrovni této hierarchické struktury je pak aplikována Saatyho metoda kvantitativního párového srovnávání. Dle těchto srovnání jsou variantám posléze přiřazeny kvantitativní charakteristiky, představující jejich důležitost. Prvky stromové struktury problému se liší dle úrovně stromu, kde čím výše je prvek umístěn, tím obecnější je prvek ve vztahu k danému problému. Typicky se tedy v nejvyšší úrovni nachází cíl vyhodnocování, v druhé úrovni kritéria vyhodnocování a ve třetí úrovni jsou posuzované varianty. Složitější úlohy mohou mít více úrovní, ve kterých jsou umístěna subkritéria nebo například hodnotitelé (pro úlohy, na jejichž hodnocení se podílí více hodnotitelů). Pro nalezení řešení je tedy nejprve zapotřebí zkonstruovat hierarchickou strukturu problému, provést párové porovnání prvků na každé úrovni struktury, a nakonec pomocí syntézy získaných preferencí zvolit nejvýhodnější varianty. [6]

3.5.5 Metody založené na minimalizaci vzdálenosti od ideální varianty

Metoda TOPSIS je založena na principu vzdálenosti od ideální a tím pádem i bazální varianty. Pro její aplikaci je vyžadováno kardinální ohodnocení variant podle kritérií a určení vah těchto kritérií.

Postup aplikace metody TOPSIS se skládá z několika kroků, přičemž nejprve je zapotřebí zkonstruovat normalizovanou matici dle vzorce

$$r_{ij} = \frac{y_{ij}}{\sqrt{\sum_{j=1}^{p} y_{ij}^2}}.$$

Sloupce výsledné matice určují vektory jednotkové délky. V dalším kroku je vypočtena normalizovaná vážená kriteriální matice vztahem

$$w_{ij} = v_j r_{ij}.$$

Následně je určena bazální varianta *D* a ideální varianta *H* na základě hodnot matice *W*, která byla sestavena v předchozím kroku. Dále jsou vypočteny vzdálenosti variant od ideální varianty

$$d_i^+ = \sqrt{\sum_{j=1}^{k} (w_{ij} - h_j)^2},$$

a také od varianty bazální

$$d_i^- = \sqrt{\sum_{j=1}^{k} (w_{ij} - d_j)^2}.$$

Nakonec jsou spočteny relativní ukazatele vzdáleností variant od varianty bazální

$$c_i = \frac{d_i^-}{d_i^+ + d_i^-},$$

přičemž hodnoty těchto ukazatelů náleží intervalu *<0; 1>*, kde hodnotu 0 nabývá bazální varianta a hodnotu 1 varianta ideální. Nakonec jsou výsledky seřazeny sestupně dle relativního ukazatele vzdálenosti od bazální varianty. Varianta, která disponuje nejdelší vzdáleností od bazální varianty, je nejvýhodnější variantou. [6]

4 NÁVRH A IMPLEMENTACE APLIKACE

Cílem praktické části práce byla implementace webové aplikace, která umožní rozhodovateli zvolit optimální bankovní účet dle jeho požadavků, a to za pomocí metod vícekriteriálního rozhodování. Aplikace mimo srovnání bankovních účtů bude také poskytovat vizualizaci nabízených bankovních účtů, včetně bankovních poplatků za jejich využívání.

Pro implementaci byl zvolen programovací jazyk C#, jelikož se jedná o preferovaný programovací jazyk autora a jde také o jeden z nejvíce rozšířených programovacích jazyků současnosti. Jako uložiště dat byla zvolena databáze MySQL, jelikož se jedná o jednoduchou databázi pro správu, instalaci a je dostupná zdarma. Další alternativa pro uložiště dat byla zvolena databáze v rámci paměti aplikace (tzv. in-memory databáze). Vývojové prostředí pro implementaci aplikace bylo zvoleno Visual Studio 2022.

Pro správu verzí projektu byl využit nástroj Git, neboť jde o nejrozšířenější nástroj pro správu verzí projektů. K hostingu a správě verzí projektu za použití nástroje Git byla zvolena služba GitHub.

Před implementací projektu bylo záměrně cíleno na použití moderních metodik pro návrh a vývoj softwaru, včetně zvolení technologií, které jsou osvědčené, využívané a moderní. Toto rozhodnutí prospěje nejen autorovi práce k prohloubení znalostí vývoje aplikací, ale také čtenáři k seznámení se s vybranými nástroji pro vývoj aplikací.

4.1 Funkční požadavky

Před samotnou implementací bylo potřeba provést sběr a analýzu funkčních a nefunkčních požadavků pro systém. Analýza požadavků by měla vést k lepšímu pochopení požadavků na systém, vymezení jeho hranic a představu o časovém plánu pro implementaci. Nejprve byl proveden sběr požadavků, přičemž tyto požadavky byly specifikovány zejména vedoucím diplomové práce. V případě funkčních požadavků se jedná zejména o požadavky na funkcionalitu aplikace:

Tabulka 1 - Funkční požadavky

Označení	**Popis požadavku**
FR01	Aplikace bude uživateli umožňovat analýzu bankovních účtů pomocí zvolených metod vícekriteriálního rozhodování.
FR02	Aplikace bude poskytovat vhodné a přehledné rozhraní pro zobrazení přehledu bankovních účtů.
FR03	Aplikace bude umožňovat vhodné zobrazení detailu bankovního účtu (název banky, název účtu, poplatky za užívání bankovního účtu, …).
FR04	Aplikace bude poskytovat přehledné zobrazení analyzovaných bankovních účtů včetně informace o konečném pořadí bankovního účtu formou tabulky.
FR05	Aplikace bude uživateli poskytovat obecnou informaci o jejím účelu a stručnou informaci o principu fungování aplikace.
FR06	Systém bude umožňovat analýzu bankovních účtů pomocí několika metod vícekriteriálního rozhodování, přičemž aplikace bude poskytovat možnost nastavení metody vícekriteriálního rozhodování, která bude použita pro analýzu.
FR07	Aplikace bude implementována jako webová.
FR08	Systém bude poskytovat REST aplikační rozhraní, pomocí něhož bude možné provádět analýzu bankovních účtů a manipulovat se zdroji systému.
FR09	Bude implementována sdílená DLL knihovna, která bude provádět generickou analýzu variant.
FR10	Funkčnost metod vícekriteriálního rozhodování budou testovány pomocí jednotkových testů, a to na několika datových sadách.
FR11	Aplikace bude uživateli poskytovat vhodné rozhraní pro vytváření, editaci a mazání dat systému.
FR12	Aplikace bude umožňovat autentizaci uživatele.
FR13	Aplikace bude umožňovat registraci uživatele.
FR14	Systém bude kalkulovat normalizované váhy kritérií pomocí metod pro stanovení vah kritérií.

Zdroj: Vlastní zpracování

4.2 Nefunkční požadavky

Níže uvedená tabulka specifikuje nefunkční požadavky aplikace, tedy doplňující požadavky architektury aplikace. Jedná se o požadavky týkající se způsobu implementace aplikace, použitých technologií a nástrojů, udržitelnost systému nebo předpoklady na výkonnost:

Tabulka 2 - Nefunkční požadavky

Označení	**Popis požadavku**
NFR01	Systém bude analyzovat bankovní účty deseti největších bank, které budou vybrány dle bilanční sumy za rok 2021.
NFR02	Systém bude analyzovat běžné a studentské bankovní účty.
NFR03	Bankovní účty budou analyzovány alespoň dle deseti kritérií.
NFR04	Bankovní účty budou analyzovány nejméně dle jednoho kvalitativního kritéria.
NFR05	Uživatelská hesla budou zabezpečena prostřednictvím dostatečně bezpečně hashovací funkce společně s přidáním soli.
NFR06	Aplikace bude responzivní.
NFR07	Aplikace bude napsána v jazyce C#.
NFR08	Data se budou ukládat do databázového uložiště MySQL.
NFR09	Analýza bankovních účtů nebude trvat déle než pět sekund.
NFR10	Systém bude implementován na .NET frameworku.
NFR11	Webový klient bude implementován jako Blazor WebAssembly aplikace.
NFR12	Bude dostupná definice REST aplikačního rozhraní prostřednictvím frameworku Swagger.
NFR13	Webová aplikace bude kompatibilní pro použití na současných verzích webových prohlížečů Google Chrome, Safari, Microsoft Edge a Mozilla Firefox.
NFR14	Verzování systému bude prováděno pomocí nástroje Git.
NFR15	Mapování objektů na entity databázového systému bude prováděno pomocí Entity Frameworku.
NFR16	Pro logování aplikace bude použit nástroj Serilog.
NFR17	Webová aplikace bude využívat Radzen UI komponenty.
NFR18	Vstup a výstup aplikačního rozhraní bude v JSON formátu.

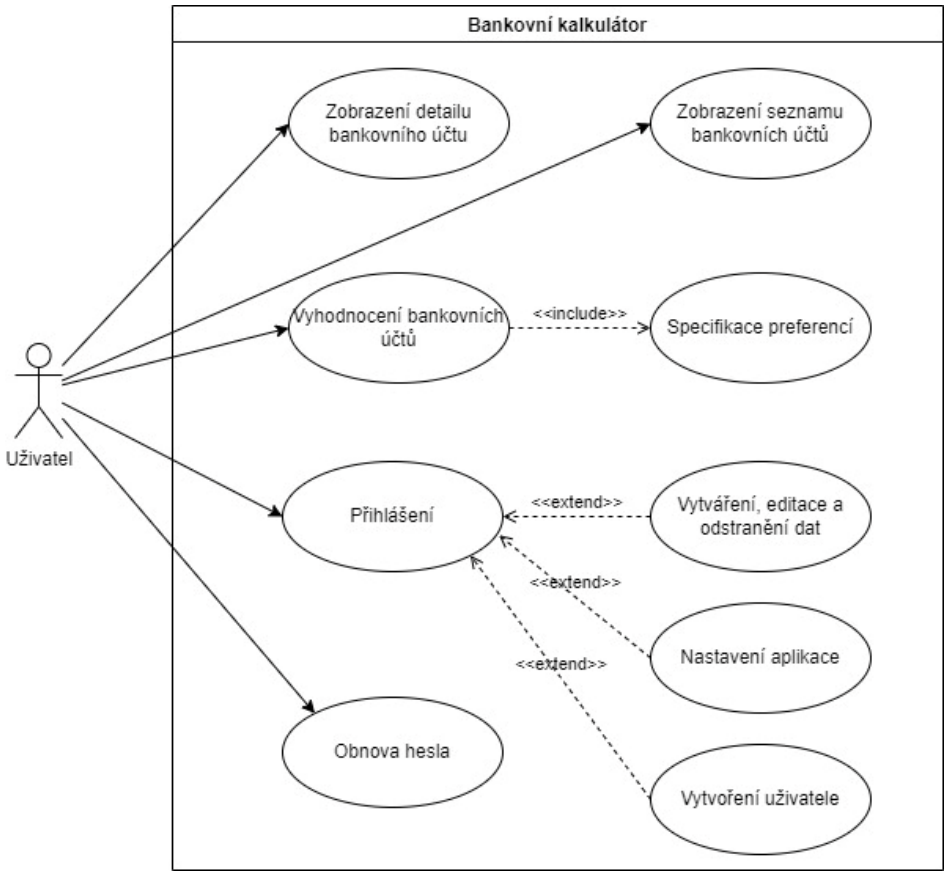
Zdroj: Vlastní zpracování

4.3 Případy užití

Pro lepší přehled je vhodné požadavky znázornit pomocí Use-Case diagramu, který znázorňuje spolupráci s koncovým uživatelem (případně jiným systémem).

S aplikací smí interagovat uživatel, přičemž jeho možnosti pro užívání aplikace jsou odvozeny od faktu, zda byl tento uživatel úspěšně autentizován. V takovém případě jsou obecné případy užití (například zobrazení dat, vyhodnocení bankovních účtů) rozšířeny o několik dalších, mezi které patří například vytváření, editace a odstraňování dat. Interakce mezi uživatelem a systémem lze znázornit následujícím diagramem případů užití:

Obrázek 1 - Use-Case diagram



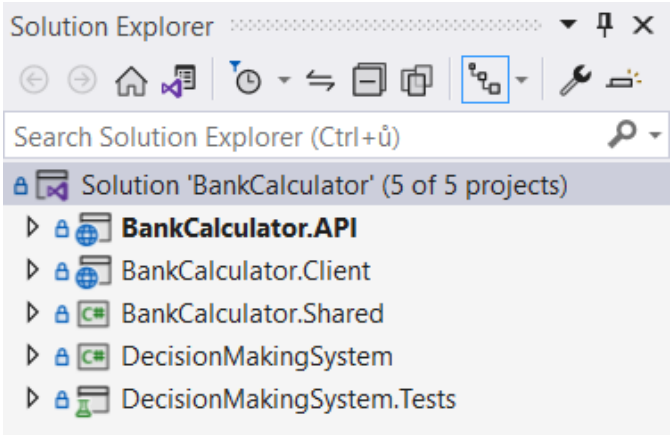
Zdroj: Vlastní zpracování

Jak již bylo zmíněno, neautorizovaný uživatel smí prohlížet seznam bankovních účtů, zobrazovat jejich detailní informace, vyhodnocovat bankovní účty a autentizovat se, což rozšíří možnosti použití aplikace uživatelem o konfiguraci aplikace, vytvoření nového uživatelského účtu a provádění CRUD operací nad daty.

4.4 Popis struktury projektu

Konečné řešení se skládá z několika projektů. Konkrétně se jedná o sdílenou knihovnu, která implementuje metody vícekriteriálního rozhodování, analýzu variant a metody pro stanovení vah kritérií. Dále byl vytvořen testovací projekt s jednotkovými testy pro testování metod vícekriteriálního rozhodování. Mezi pravděpodobně nejvýznamnější moduly konečného řešení patří projekt webového aplikačního rozhraní a webového klienta. Dalším vytvořeným projektem je sdílená knihovna, která definuje objekty pro přenos dat mezi aplikačním rozhraním a webovým klientem. Projekty konečného řešení lze vidět na obrázku níže:

Obrázek 2 - Struktura projektu



Zdroj: Vlastní zpracování

4.4.1 Sdílená knihovna pro vyhodnocení variant

Knihovna implementuje několik metod vícekriteriálního rozhodování a metod pro stanovení vah kritérií. Dále poskytuje třídy reprezentující matici variant, kritéria, jejich typ, ohodnocenou variantu atd. Jedním z důvodů oddělení implementace těchto tříd do samostatné knihovny bylo mimo jiné testování metod vícekriteriálního rozhodování pomocí jednotkových testů. Testování metod vícekriteriálního rozhodování se vytvořením samostatné knihovny výrazně zjednodušilo, jelikož je možné testovat tyto metody bez ovlivnění výsledků testů dalšími aspekty.

Důvodem implementace vícera metod vícekriteriálního rozhodování bylo, že v době implementace nebylo jisté, která metoda bude nejvhodnější pro analýzu bankovních účtů, jelikož princip funkčnosti a předpoklady použití metod se odlišují. Nejpodstatnější odlišností je pak zejména vliv vah kritérií na určení nejvýhodnějších variant. Z tohoto důvodu byly implementovány tyto metody vícekriteriálního rozhodování:

- metoda bazické varianty
- Lexikografická metoda
- metoda pořadí
- metoda TOPSIS
- metoda vážených součtů

Zároveň byly implementovány dvě metody pro stanovení vah kritérií – a sice metoda bodovací a Saatyho metoda. Důvodem k implementaci více metod pro stanovení vah kritérií bylo použití takové metody, která bude vhodná pro určení vah vzhledem k parametrům obdržených z formuláře klientské aplikace.

Některé ze zvolených metod bylo potřeba modifikovat tak, aby co nejdůvěryhodněji vyhodnocovaly varianty, případně váhy kritérií. Z metod pro vícekriteriální rozhodování tak bylo potřeba upravit vyhodnocení variant pomocí Lexikografické metody, jelikož výstup této metody neposkytuje ohodnocení každé varianty ze vstupní množiny variant, ale pouze omezený počet vyhovujících variant. Proces vyhodnocení variant pomocí této metody bylo tedy zapotřebí upravit tak, aby byla vyhodnocena každá varianta ze vstupní množiny.

4.4.2 Jednotkové testy

Pro testování metod vícekriteriálního rozhodování byly využity jednotkové testy – tedy typ testů k ověření funkčnosti malých samostatných jednotek systému. Otestování malých jednotek systému pak poskytuje podrobnější přehled o funkčnosti systému a lze odhalit nedostatky v implementaci systému a následně tyto nedostatky odstranit.

Pro testování jednotek systému byl použit zdarma dostupný nástroj xUnit, který poskytuje vysokou flexibilitu a jednoduché použití. Podporuje mimo jiné i integrační nebo selenium testy. Základními atributy nástroje jsou Fact, Theory a Skip. Atribut Fact deklaruje testovací metodu. Atribut Theory představuje sadu testů, které provádějí stejný kód s odlišnými vstupními daty. Tyto typy testovacích metod lze také popsat tak, že „fakt“ je typ testu, jehož výsledek je vždy booleovská hodnota „true“ a testují podmínky běhu programu, naopak „teorie“ je typ testu,

jehož výsledkem je hodnota „true“ pouze pro určitou množinu dat. Třetí z uvedených atributů Skip umožňuje vynechat provedení testu, přičemž je jeho vynechání potřeba zdůvodnit komentářem. Níže lze vidět ukázku několika jednotkových testů pro ověření funkčnosti analýzy variant pomocí metody váženého součtu:

Obrázek 3 - Ukázka jednotkových testů

```
41      [Fact]
        0 references
42      public void SolveByWeightedSumMethod()
43      {
44          var weightedSumMethod = new WeightedSumMethod<string>();
45          var result = weightedSumMethod.Process(_variantMatrix);
46          Assert.True(result.VariantEvaluations.Count == 5);
47          Assert.Equal("Tablet 1", result.VariantEvaluations.First().Variant.Identifier);
48      }
49
50      [Theory, MemberData(nameof(SolveDecisionTaskData))]
        0 references
51      public void SolveByWeightedSumMethod_EmptySolution(VariantMatrix<string> variantMatrix, int resultCount)
52      {
53          var weightedSumMethod = new WeightedSumMethod<string>();
54          Assert.Equal(weightedSumMethod.Process(variantMatrix).VariantEvaluations.Count, resultCount);
55      }
56
57      [Fact]
        0 references
58      public void SolveByWeightSumMethod_ShouldThrowException()
59      {
60          var weightedSumMethod = new WeightedSumMethod<string>();
61          VariantMatrix<string> variantMatrix = null;
62          Assert.Throws<ArgumentNullException>(() => weightedSumMethod.Process(variantMatrix));
63      }
```

Zdroj: Vlastní zpracování

Pro otestování metod vícekriteriálního rozhodování byla použita data vztažená k bankovním účtům a data z použité literatury. Důležitá byla zejména data z literatury vzhledem ke skutečnosti, že výsledky vyhodnocení variant jsou předem známé. Jednotkový test tak musí skončit se stejným výsledkem jako výpočet uvedený v literatuře. [7]

4.4.3 REST aplikační rozhraní

Aplikační rozhraní (API) je souhrn pravidel, který definuje, jakým způsobem aplikace navzájem komunikují. REST API je webové aplikační rozhraní, jehož zdroje (data, případně další funkcionality) jsou zpřístupňovány prostřednictvím URI, což je identifikátor, který jednoznačně určuje zdroj. Vytvoření a vystavení API rozhraní umožňuje zpřístupnění zdrojů aplikacím, nezávisle na způsobu implementace klienta. Pod názvem „klient“ si lze představit osobu či systém, který přistupuje k datům prostřednictvím tohoto rozhraní. Vzhledem k popularitě REST rozhraní a jeho výhodám, které budou v této kapitole popsány, bylo zvoleno právě toto řešení k implementaci rozhraní jakožto serverové části aplikace. [8]

4.4.3.1 Obecný popis REST API

REST API je typ komunikačního rozhraní, které umožňuje jednoduchý přístup a manipulaci se zdroji. Aplikace, která ke zdrojům přistupuje, se nazývá klient. REST rozhraní mohou být implementována pomocí různých programovacích jazyků a podporují různé datové formáty.

S REST rozhraním lze komunikovat pomocí HTTP requestu a provádět základní operace se zdroji (čtení, vytváření, editace, odstranění), kde požadovaný zdroj je určen identifikátorem URI a způsob manipulace se zdrojem je závislý na použité HTTP metodě. Odpověď je nejčastěji odeslána ve formátu JSON, který je snadno čitelný pro člověka i stroj. Odpověď kromě dat obsahuje také takzvané hlavičky, které obsahují například metadata nebo autorizační údaje.

Se zdroji lze manipulovat pomocí čtyř metod, které se často označují zkratkou CRUD (create, read, update, delete). Konkrétně se jedná o metody HTTP protokolu – POST, GET, PUT a DELETE. Účel použití většiny z těchto metod lze odvodit z jejich názvu. Metoda POST slouží pro vytvoření zdroje, metoda GET se využívá k dotazování na zdroj, metody PUT a DELETE pro editaci, respektive odstranění.

Požadavek (request) pro REST rozhraní se skládá z několika částí:

- URI – jednoznačný identifikátor zdroje. REST služby většinou identifikují zdroje pomocí URL, která určuje cestu ke zdroji. Jedná se o adresu podobnou jako často využívané adresy pro přístup k webovým stránkám. Tato adresa se také často označuje jako „endpoint“ a mimo jiné také určuje operaci, kterou klient vyžaduje provést.
- Metoda HTTP protokolu – určuje serveru, jakým způsobem klient chce manipulovat s daty. Běžně využívané metody jsou POST, GET, PUT, DELETE.
- HTTP hlavička – metadata zasílaná mezi serverem a klientem. V hlavičkách se často nachází informace například o formátu požadavku a odpovědi nebo autorizační údaje.
- Data – požadavek může obsahovat data pro korektní práci se zdroji. Data jsou nejčastěji vyžadována v případě zaslání požadavků pro vytvoření a editaci zdroje.
- Parametry – požadavek může být zaslán s dalšími parametry, které serveru poskytují více informací o tom, jak požadavek zpracovat Parametry lze předat v rámci cesty ke zdroji (URL) nebo lze zaslat takzvané query parametry, pomocí nichž lze například filtrovat zdroje.

Odpověď se zpravidla skládá z těchto částí:

- Status kód – třímístné číslo, které oznamuje o úspěšnosti zpracování požadavku. Status kód začínající číslicí 2 indikuje úspěšné zpracování požadavku. Naopak stavový kód začínající číslicí 4 nebo 5 indikuje chybné zpracování. Pokud status kód začíná číslicí 3, pak došlo k přesměrování URL. Jako příklad lze uvést status kódy 200 (úspěšné zpracování požadavku), 201 (úspěšné vytvoření zdroje), 400 (chybné zpracování – předpoklad chyby klienta), 404 (server nemůže nalézt požadovaný zdroj) nebo 500 (neočekávaná chyba serveru).
- Tělo odpovědi – obsahuje reprezentaci zdroje. Nejčastěji reprezentováno v JSON formátu.
- Hlavičky – obsahují metadata odpovědi, jako například datum a čas zpracování požadavku, kódování nebo formát dat odpovědi.

REST rozhraní musí splňovat následující kritéria:

- Jednotné rozhraní – všechny požadavky pro identický zdroj by měly mít stejnou formu. REST aplikační rozhraní by navíc mělo zajistit jednoznačnou identifikaci zdroje. Odpověď pro klienta by měla obsahovat pouze data, která jsou od klienta vyžadována.
- Oddělení odpovědnosti mezi klientem a serverem – aplikace klienta a serveru musejí být navzájem nezávislé. Jediná informace, kterou by měl klient znát, je URI požadovaného zdroje. Klient by neměl být schopen interagovat s rozhraním jakýkoli jiným způsobem.
- Bezstavovost – mezi klientem a rozhraním není udržováno spojení a každý požadavek tak musí obsahovat všechny informace potřebné pro jeho zpracování.
- Cacheability – je možné na straně klienta nebo serveru ukládat zdroje v mezipaměti a zvýšit tak výkonnost systému. Odpověď serveru by v případě ukládání zdrojů do mezipaměti měla obsahovat informaci, zda je tato možnost pro zdroj povolena.
- Vícevrstvá architektura – je potřeba brát v potaz, že klientská a serverová aplikace spolu nekomunikují přímo, a tak je nutné počítat s možnými prostředníky. REST API je zapotřebí navrhnout tak, aby klient ani server nemohl vědět, zda komunikuje s koncovou aplikací nebo prostředníkem.

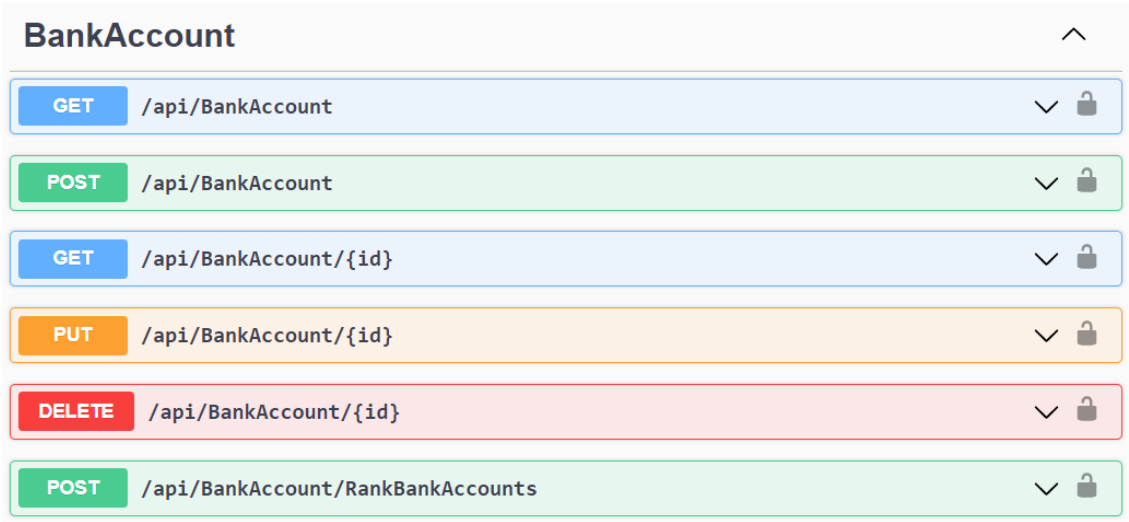
- Code on demand – volitelná vlastnost. Servery mohou (spíše výjimečně) upravit funkčnost klienta zasláním spustitelného kódu v odpovědi.

Ačkoli rozhraní musí splňovat tyto požadavky, tak i přesto je jeho použití považováno za jednodušší něž například aplikační rozhraní, které pro komunikaci využívá protokol SOAP, což je také jeden z typů webového rozhraní, který zasílá zprávy ve formátu XML. Toto rozhraní má vestavěné zabezpečení a transakční mechanismy, což znamená, že oproti REST rozhraním je pomalejší, méně škálovatelné a náročnější na systémové prostředky. [8], [9]

4.4.3.2 Popis aplikačního rozhraní aplikace

Aplikační rozhraní aplikace je založeno na .NET 6 frameworku. Tato část aplikace poskytuje rozhraní pro přístup a manipulaci s daty, vyhodnocuje bankovní účty pomocí metod vícekriteriálního rozhodování, autorizuje uživatele a vytváří prostředí pro ukládání dat aplikace. Pomocí tohoto rozhraní lze vytvářet, editovat a odstraňovat zdroje. Jedná se především o objekty představující bankovní účty a banky. Dále lze díky tomuto rozhraní editovat nastavení – konkrétně metodu vícekriteriálního rozhodování, která bude využita pro vyhodnocení bankovních účtů.

Obrázek 4 - Ukázka endpointů aplikačního rozhraní aplikace



Zdroj: Vlastní zpracování

Koncové body pro editaci dat je potřeba zabezpečit tak, aby na ně nemohl zasílat požadavky neautorizovaný klient. Z toho důvodu navíc rozhraní poskytuje endpoint pro autorizaci uživatele prostřednictvím protokolu OAuth 2.0.

4.4.3.3 Autorizace pomocí OAuth 2 protokolu

OAuth 2.0 je autorizační protokol, který slouží k umožnění přístupu ke zdrojům, které jsou hostované jinými aplikacemi. Pro autorizaci se využívají takzvané „access tokeny“, což jsou data odeslaná uživatelem, která umožňují klientské aplikaci přístup k zabezpečenému zdroji.

Ačkoli není přesně definovaný formát pro tyto tokeny, nejvyužívanější jsou takzvané JWT (JSON Web Token). Jedná se o otevřený standard, který definuje kompaktní a zabezpečený způsob přenosu informací mezi stranami v JSON formátu. Těmto informacím lze důvěřovat a ověřit je, jelikož jsou digitálně podepsané. Pomocí tohoto podpisu lze ověřit, že token nebyl nijak upravený od jeho vytvoření. Podepsána mohou být tajným klíčem za použití algoritmu HMAC, nebo párem privátního a veřejného klíče pomocí šifrovacího algoritmu RSA nebo ECDSA.

JWT token se skládá ze tří částí – hlavičky, datové části a podpisu. Jednotlivé části odděluje tečka. Hlavička a datová část jsou kódované prostřednictvím Base64. Hlavička má obvykle dvě položky – a sice položku popisující podpisový algoritmus a informaci o typu tokenu (většinou se jedná o hodnotu JWT). Datová část obsahuje „prohlášení“ (takzvané claims) a případná další libovolná data. Jako příklad prohlášení lze zmínit například informaci o expiračním čase, vystaviteli tokenu, nebo o času jeho vytvoření. Podpisová část se vytvoří tím způsobem, že se pomocí kódování Base64 zakóduje hlavička, datová část, připojí se tajný klíč a všechna tato data se digitálně podepíší. Spojením těchto tří částí vznikne JWT token, který lze snadno přenášet prostřednictvím HTTP protokolu. [10], [11]

Na následujícím obrázku lze vidět příklad JWT tokenu, který byl vytvořen autorizačním dotazem na endpoint aplikačního rozhraní systému (použité barvy znázorňují jednotlivé části tokenu):

Obrázek 5 - Příklad access tokenu

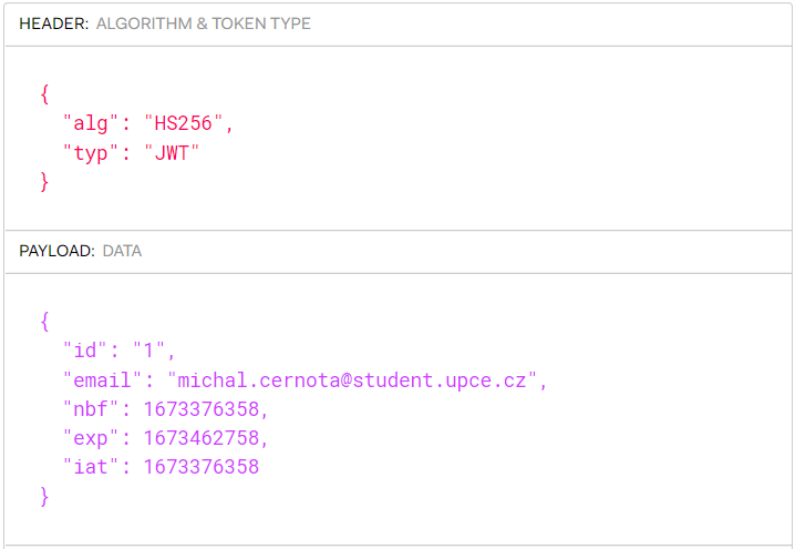
```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JpZCI6IjEiLCJlbWFpbCI6Im1pY2hhbC5jZXJub
3RhQHN0dWRlbnQudXBjZS5jeiIsIm5iZiI6MTY3
MzM3NjM1OCwiZXhwIjoxNjczNDYyNzU4LCJpYXQ
iOjE2NzMzNzYzNTh9.K0-
bEh68X2aLJKuBpfzUb5ufm5NQIasJ4Cq2gEUEO5
o
```

Zdroj: Vlastní zpracování

Na obrázku níže lze vidět již dekódované části tokenu. Hlavička obsahuje informaci o typu tokenu a použitého algoritmu pro podpis. Hodnoty datové části tokenu mají následující význam:

- id – unikátní identifikátor uživatele
- email – email uživatele
- nbf – časová položka určující počátek validity tokenu („not valid before“)
- exp – časová položka určující čas expirace tokenu („expiration time“)
- iat – časová položka určující čas vytvoření tokenu („issued at“)

Obrázek 6 - Dekódovaný access token



Zdroj: Vlastní zpracování

4.4.3.4 Objektově relační mapování

Aplikační rozhraní aplikace využívá techniku automatické konverze dat mezi relační databází a objekty konkrétního programovacího jazyka. Tato technika, která je označována jako ORM, poskytuje vývojáři možnost snadného provádění běžných databázových operací, bez nutnosti dotazování se SQL dotazy, a zajišťuje tak abstrakci od databázového systému. Mimo to zajišťuje automatickou konverzi datových typů a persistenci dat mezi objekty používanými v aplikaci a jejich reprezentací v databázovém systému. Další velmi častou vlastností ORM je nezávislost na typu databázového systému, což umožňuje zvolit databázový systém dle požadavku uživatele.

Pro objektově relační mapování byl v aplikaci použit rámec Entity Framework. Ten umožňuje využití nejen standardních vlastností ORM nástroje, ale také dotazování pomocí integrovaného jazyk LINQ, použití transakcí, nebo například umožňuje ukládání dat do mezipaměti.

Existují dva přístupy vývoje, co se týká objektově relačního mapování. Prvním z nich je přístup „database-first“, kdy je již vytvořený databázový model a doménové třídy jsou následně generovány dle definice databázových tabulek. Druhý přístup je takzvaný „code-first“, kdy jsou naopak definovány doménové třídy a dle nich je vytvořeno databázové schéma. Jelikož nebyla předem vytvořena databáze, byl zvolen druhý přístup – byly tedy nejdříve definovány doménové (entitní) třídy, které byly posléze transformovány na databázové tabulky.

Entitním třídám bylo potřeba určit vlastnosti, jako například zda určitý atribut třídy má být primárním klíčem tabulky. Entity Framework toto umožňuje provést vícero způsoby. Dále umožňuje určit názvy sloupců, jejich datový typ, omezení, primární a cizí klíče. Některé charakteristiky jsou určeny implicitně. Primárním klíčem entity se například stává automaticky atribut (v případě .NET frameworku vlastnost) s názvem „Id“. Dalšími způsoby určení databázových sloupců a tabulek je buď pomocí anotací nebo pomocí takzvaného Fluent API. Ačkoli se použití anotací zdá být přímočařejší, tak pro specifikaci databáze bylo zvoleno Fluent API, díky čemuž jsou doménové třídy přehlednější a čistší. Na následujícím obrázku lze vidět příklad definice tabulky určené pro banky pomocí Fluent API: [12]

Obrázek 7 - Konfigurace a mapování vlastností

```
15 references
public class AppDbContext : DbContext
{
    2 references
    public DbSet<Bank> Banks { get; set; }
    5 references
    public DbSet<BankAccount> BankAccounts { get; set; }
    0 references
    public DbSet<AccountInfoFees> AccountInfoFees { get; set; }
    0 references
    public DbSet<AccountManagementFees> AccountManagementFees { get; set; }
    0 references
    public DbSet<AccountPaymentFees> AccountPaymentFees { get; set; }
    0 references
    public DbSet<AccountWithdrawalFees> AccountWithdrawalFees { get; set; }
    4 references
    public DbSet<User> Users { get; set; }

    0 references
    public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Bank>().ToTable("Bank");
        modelBuilder.Entity<Bank>().HasKey(bank => bank.Id);
        modelBuilder.Entity<Bank>().Property(bank => bank.Id).ValueGeneratedOnAdd();
        modelBuilder.Entity<Bank>().Property(bank => bank.Name).IsRequired().HasMaxLength(300);
        modelBuilder.Entity<Bank>().Property(bank => bank.NetProfitInBillions).IsRequired();
        modelBuilder.Entity<Bank>().Property(bank => bank.NumberOfAtms).IsRequired();
        modelBuilder.Entity<Bank>().Property(bank => bank.NumberOfClientsInThousands).IsRequired();
        modelBuilder.Entity<Bank>().Property(bank => bank.OperatingSince).IsRequired();
        modelBuilder.Entity<Bank>().Property(bank => bank.BalanceSheetInBillions).IsRequired();
```

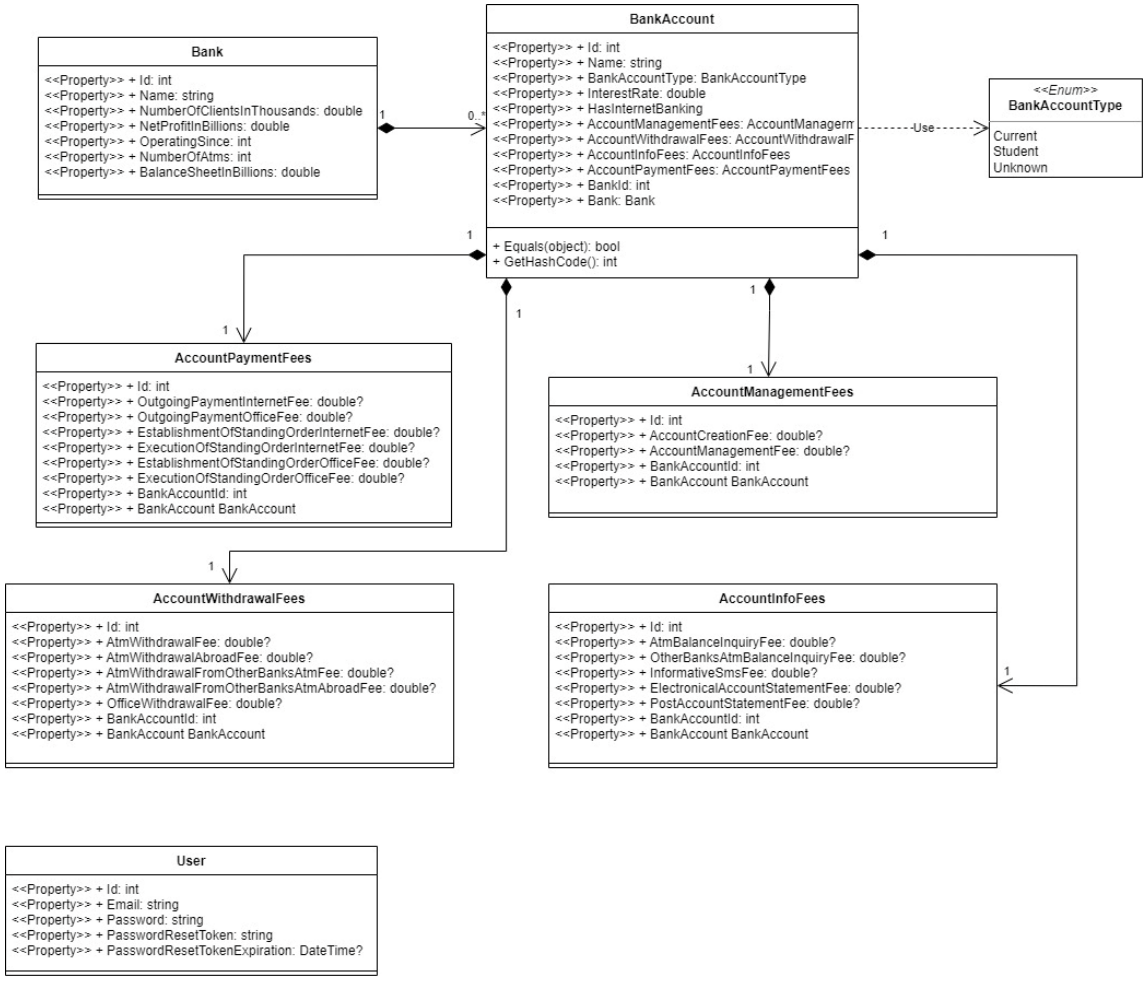
Zdroj: Vlastní zpracování

Na obrázku je možné vidět určení primárního klíče tabulky, automatické generování hodnoty primárního klíče při vytvoření záznamu a určení omezení hodnot (maximální délka řetězce a povinné hodnoty).

Aplikace ukládá data do celkem osmi tabulek. Primárně se jedná o tabulky, které uchovávají data o bankách, jejich bankovních účtech a poplatcích těchto bankovních účtů. Další databázová tabulka je určena pro ukládání uživatelských účtů.

Doménové objekty, které jsou mapovány na entity relační databáze, znázorňuje diagram tříd níže. Za zmínění stojí především fakt, že sazby poplatků byly separovány do samostatných tříd. Toto rozhodnutí bylo učiněno z důvodu vysokého počtu atributů databázové tabulky, která obsahuje informace o bankovních účtech. Veškeré sazby poplatků musejí umožňovat uchování hodnot null. Některé bankovní účty totiž nemají v ceníku určené sazby vybraných poplatků nebo některé ze služeb, které nabízejí jiné bankovní účty, svým uživatelům nenabízejí.

Obrázek 8 - Diagram doménových tříd



Zdroj: Vlastní zpracování

4.4.3.5 Vrstvy aplikačního rozhraní

Základní strukturu aplikace lze popsat jako několik vrstev s odlišnou odpovědností. Byla použita architektura Controller-Service-Repository, která se v současnosti běžně používá pro implementaci aplikačních rozhraní nejen v .NET frameworku. Controller implementuje operace definované aplikačním rozhraním aplikace, určuje HTTP metodu endpointu, popisuje návratové datové typy, status kódy atd. Třídy služeb představují vrstvu implementující business logiku aplikace. Repositáře jsou objekty, které zajišťují abstrakci od datové vrstvy a poskytují operace pro provádění CRUD operací v datovém uložišti. Byla vytvořena abstraktní generická třída Repository<TEntity>, která implementuje rozhraní IRepository<TEntity>. Toto rozhraní určuje, že odvozené repositáře budou poskytovat základní operace pro manipulaci s daty. Tímto krokem byl taktéž eliminován duplicitní kód. Na následujícím obrázku lze vidět implementaci abstraktní třídy repositáře:

Obrázek 9 - Abstraktní třída repositáře

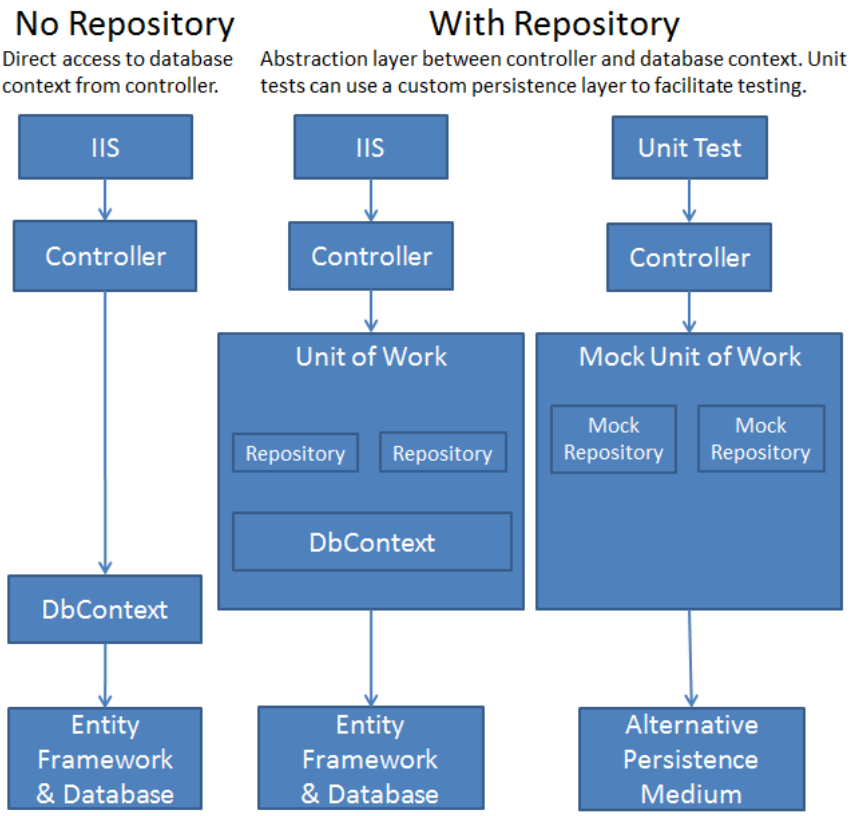
```
7 references
public abstract class Repository<TEntity> : IRepository<TEntity> where TEntity : class
{
    protected readonly AppDbContext _context;

    3 references
    public Repository(AppDbContext context)
    {
        _context = context;
    }

    3 references
    public void Delete(TEntity entity)
    {
        _context.Set<TEntity>().Remove(entity);
    }

    5 references
    public IEnumerable<TEntity> GetAll()
    {
        return _context.Set<TEntity>().ToList();
    }

    9 references
    public TEntity GetById(int id)
    {
        return _context.Set<TEntity>().Find(id);
    }

    4 references
    public void Add(TEntity entity)
    {
        _context.Add(entity);
    }
```

Zdroj: Vlastní zpracování

Dále byl využit takzvaný „Unit of Work“ pattern, který zajišťuje, že sekvence na sebe navazujících operací nad daty bude provedena jako atomická operace, která se buď provede

celá, nebo v případě chyby budou data vrácena do původního stavu, čímž bude zajištěna konzistence dat. To je zajištěno tím, že všechny repositáře sdílí stejnou instanci třídy AppDbContext, která je zodpovědná za interakci s datovým uložištěm. Třída reprezentující vzor „Unit of Work“ tedy zapouzdřuje všechny repositáře pro entity. Tuto architekturu lze znázornit následujícím diagramem:

Obrázek 10 - Unit of Work pattern



Zdroj: [41]

4.4.3.6 Konfigurace aplikace

Aplikaci lze konfigurovat editací souboru appsettings.json. V tomto souboru lze konfigurovat logování aplikace do souboru pomocí nástroje Serilog, filtraci příchozích požadavků nebo použitou metodu pro analýzu variant, kterou je však doporučeno editovat buď pomocí odeslaného requestu na aplikační rozhraní nebo v klientské aplikaci, a to z důvodu předejití chybné konfigurace aplikace.

Nejdůležitější částí konfiguračního souboru aplikace je bezesporu konfigurace datového uložiště. Aplikace podporuje MySQL databázové uložiště nebo ukládání dat do databáze, která je vytvořena v paměti aplikace (tzv. in-memory databáze). Pokud bude jako datové uložiště využívána MySQL databáze, je zapotřebí nastavit atribut „UseInMemoryDatabase“ na hodnotu

„false“ a také nastavit connection string pro připojení k databázi. V opačném případě postačí upravit atribut „UseInMemoryDatabase“ na hodnotu „true“. Ukládání dat v rámci in-memory databáze je doporučeno pouze pro vývoj aplikace a pro testovací účely. Konfigurační soubor mimo jiné také uchovává symetrický klíč pro podpis JWT tokenu. Na následujícím obrázku lze vidět obsah konfiguračního souboru aplikace:

Obrázek 11 - Konfigurační soubor aplikace

```
  },
  "Serilog": {
    "Using": [ "Serilog.Sinks.File", "Serilog.Sinks.Console" ],
    "MinimumLevel": {
      "Default": "Information"
    },
    "WriteTo": [
      {
        "Name": "File",
        "Args": {
          "path": "logs/webapi-.log",
          "rollingInterval": "Day",
          "outputTemplate": "[{Timestamp:yyyy-MM-dd HH:mm:ss.fff zzz} {CorrelationId} {Level:u3}] {Username} {Message:lj}{NewLine}{Excepti
          "encoding": "System.Text.Encoding::UTF8"
        }
      },
      {
        "Name": "Console"
      }
    ]
  },
  "Security": {
    "Secret": "h0SHehONcezbFqDTeKrtHLbc0p0NAyzAjdczSNiorWmLjqMXQF"
  },
  "DecisionAnalysis": {
    "DecisionMethodType": "WeightedSumMethod"
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "Dev": "server=localhost;port=3306;database=bank;uid=root;password=root"
  },
  "UseInMemoryDatabase": true
}

```

Zdroj: Vlastní zpracování

Z ukázky konfiguračního souboru lze mimo jiné zjistit, že aplikace ukládá logovací soubory do podadresáře „log“, který se nachází v hlavním adresáři aplikace. Dále je určena šablona pro záznam v logovacím adresáři, interval vytvoření nového logovacího souboru nebo použité kódování znaků.

4.4.4 Webový klient

Další nezbytnou částí aplikace je webový klient. Jedná se o aplikaci s grafickým rozhraním, která má více účelů. Hlavním účelem webového klienta je poskytovat uživateli vhodný způsob zadání preferencí pro vyhodnocení bankovních účtů a zobrazení výsledků hodnocení variant – tedy doporučené bankovní účty. Mimo to bude aplikace umožňovat zobrazení všech bankovních účtů bank, které byly zvoleny pro analýzu, jejich podrobné informace a obecné informace aplikace se stručným uvedením do problematiky vícekriteriálního rozhodování. Dále bude poskytovat vhodné rozhraní pro editaci dat a také formulář pro přihlášení uživatele.

Pro implementaci webového klienta byl zvolen bezplatný webový rámec vyvíjený společností Microsoft – Blazor. Jedná se o framework s otevřeným zdrojovým kódem pro vývoj jednostránkových aplikací. Jednostránková aplikace (SPA) je typ webové aplikace, která dynamicky modifikuje fragmenty jediné webové stránky po interakci uživatele, oproti generování celé webové stránky, což je výchozí chování webového prohlížeče. Hlavní výhodou tohoto přístupu je výrazně rychlejší interakce aplikace.

Jedním z hlavních důvodů, proč pro implementaci webového klienta byl zvolen právě Blazor je, že aplikace je vytvářena (primárně) programovacím jazykem C# a značkovacím jazykem HTML. Jelikož C# je preferovaným programovacím jazykem autora práce, tak pro implementaci klienta byla zvolena právě tato technologie na úkor zvolení javascriptové knihovny React, která byla jednou z dalších mnoha možností.

4.4.4.1 Hostovací modely Blazoru

Blazor v současné době nabízí dvě varianty implementace webových klientů. První z těchto variant je takzvaný „Blazor Server“, který byl představen v roce 2018. Druhou možností implementace klienta je „Blazor WebAssembly“, která byla představena o rok později. Liší se zejména ve způsobu hostování Razor komponent, které lze popsat jako základní jednotky Blazor aplikace (například tabulka, tlačítko).

Blazor Server poskytuje podporu pro hostování těchto komponent na serveru v ASP.NET Core aplikaci, přičemž aktualizace uživatelského rozhraní se zpracovávají prostřednictvím komunikace SignalR, což je knihovna, která usnadňuje přidávání webových funkcí aplikacím v reálném čase. Runtime tedy zůstává na straně serveru a připojení je se serverem navázáno ze strany klienta. Z toho vyplývá, že aplikace na straně klienta je zodpovědná za persistenci a případnou obnovu stavu aplikace dle potřeby. Nevýhodou může být vyšší latence nebo nefunkčnost aplikace v případě, že dojde například k chybě komunikace se serverem.

Blazor WebAssembly (zkráceně WASM) umožňuje spuštění .NET kódu přímo ve webovém prohlížeči. WebAssembly je kompaktní formát byte kódu, který je optimalizovaný pro rychlé stahování a maximální rychlost provádění. Jedná se o webový standard, a tak je WebAssembly podporováno ve všech moderních prohlížečích (včetně mobilních), a to bez potřeby instalace dodatečných pluginů. Kód WebAssembly může přistupovat k veškeré funkcionalitě prohlížeče prostřednictvím JavaScriptu. Klíčovým faktorem je pak velikost aplikace. Jelikož je celé sestavení aplikace staženo do prohlížeče, tak pokud je aplikace velmi komplexní, tak její spuštění může být pomalejší. Po stažení veškerých potřebných prostředků do prohlížeče je však zpravidla rychlost aplikace vyšší oproti Blazor Serveru, který pro svou funkčnost vyžaduje

komunikaci se serverem. Postup sestavení a spuštění aplikace Blazor WebAssembly pak lze popsat následovně:

1) C# kód a Razor komponenty jsou kompilovány do .NET sestavení (assemblies)
2) Zkompilovaná aplikace, závislosti a běhové prostředí (runtime) jsou staženy do prohlížeče
3) Blazor WebAssembly zavede běhové prostředí .NET a nakonfiguruje jej pro načtení sestavení aplikace
4) Blazor WebAssembly využívá JavaScript pro manipulaci s objektovým modelem dokumentu (Document Object Model – zkráceně DOM) a pro interakci s rozhraním webového prohlížeče

Jelikož platí předpoklad, že webový klient nebude příliš komplexní, byla zvolena implementace webového klienta jakožto aplikace Blazor WebAssembly, a to i za cenu jejího pomalejšího spuštění. Výhodou je, že není potřeba serverová aplikace pro práci s komponentami, což urychlí vývoj aplikace. [13]

4.4.4.2 Implementace webového klienta

Jak již bylo zmíněno, tak webový klient byl implementován jako aplikace Blazor WebAssembly. V následující kapitole budou popsány hlavní funkcionality aplikace a způsob její implementace.

Pro rychlejší vývoj klienta s cílem vytvoření vizuálně přívětivého a moderního grafického rozhraní byla využita bezplatně dostupná knihovna komponent s názvem Radzen. Jedná se o knihovnu komponent pro Blazor aplikace, která obsahuje více než 70 různých komponent, jež jsou podporovány jak pro aplikace Blazor WebAssembly, tak pro aplikace Blazor Server. Z dostupných komponent lze zmínit například tabulky, ovládací prvky pro formuláře, grafy, nabídky nebo tlačítka. Použitím této knihovny se tak velmi urychlil vývoj, jelikož nebylo nutné tyto komponenty dodatečně implementovat a ani pro ně vytvářet kaskádové styly.

Na následujícím obrázku lze vidět ukázku použití Radzen komponenty pro zobrazení bankovních účtů na stránce aplikace:

Obrázek 12 - Ukázka implementace stránky Blazor aplikace

```
@page "/bank-accounts";
@using BankCalculator.Client.Service;
@using BankCalculator.Client.Service.Interface;
@using BankCalculator.Shared.DTOs.BankAccount
@inject HttpClient HttpClient;
@inject IHttpService HttpService;
@inject NavigationManager Nav;
@inject IAuthService AuthenticationService;

<RadzenText TextStyle=TextStyle.DisplayH4>Bankovní účty</RadzenText>

@if (_bankAccounts == null)
{
    <LoadingSpinner />
}
else
{
    <RadzenButton ButtonStyle="ButtonStyle.Success" Icon="add_circle_outline" class="mt-2 mb-4"
        Text="Vytvořit účet" Click="@InsertRow" Visible=@AuthenticationService.Authorized/>
    <RadzenDataGrid @ref=dataGrid Data="@_bankAccounts" TItem="BankAccountDetailDto" PageSize="15" ShowMultiColumnSortingIndex="true"
                AllowPaging="true" AllowSorting="true" SelectionMode="DataGridSelectionMode.Single" AllowRowSelectOnRowClick=true
                RowSelect="@RowSelect">
        <Columns>
            <RadzenDataGridColumn TItem="BankAccountDetailDto" Property="Id" Visible=false />
            <RadzenDataGridColumn TItem="BankAccountDetailDto" Property="Bank.Name" Title="Banka" Width="160px" />
            <RadzenDataGridColumn TItem="BankAccountDetailDto" Property="Name" Title="Název účtu" Width="160px" />
            <RadzenDataGridColumn TItem="BankAccountDetailDto" Property="BankAccountType" Title="Typ účtu" Width="160px" />
            <RadzenDataGridColumn TItem="BankAccountDetailDto" Context="bankAccount" Filterable="false" Sortable="false"
                Visible=@AuthenticationService.Authorized TextAlign="TextAlign.Right" Width="156px">
                <Template Context="bankAccount">
                    <RadzenButton Icon="edit" ButtonStyle="ButtonStyle.Light" Variant="Variant.Flat" Size="ButtonSize.Medium"
                        Click="@(args => EditRow(bankAccount))" @onclick:stopPropagation="true" />
                    <RadzenButton ButtonStyle="ButtonStyle.Danger" Icon="delete" Variant="Variant.Flat" Shade="Shade.Lighter"
                        Size="ButtonSize.Medium" class="my-1 ms-1" Click="@(args => DeleteRow(bankAccount))" @onclick:stopPropagation="true" />
                </Template>
            </RadzenDataGridColumn>
        </Columns>
    </RadzenDataGrid>
}
```

Zdroj: Vlastní zpracování

Na ukázce výše lze kromě použití grafické komponenty pro zobrazení tabulky vidět direktivy a atributy charakteristické pro implementaci Blazor aplikací. Direktiva @page určuje, že se jedná o směrovatelnou komponentu, která může být zpřístupněna požadavkem uživatele po zadání URL adresy. Dále lze vidět příklad využití techniky vkládání závislostí (Dependency Injection) do komponenty pomocí @inject direktivy. Na ukázce lze také vidět podmíněné vykreslení komponenty, kde se tabulka, která zobrazuje bankovní účty, zobrazí až po načtení dat. To je provedeno po zpracování odpovědi na request, který je po načtení stránky odeslán na aplikační rozhraní aplikace.

Obrázek č. 13 zobrazuje další část zdrojového kódu stránky. Jedná se o direktivu @code, ve které lze implementovat standardní prvky jazyka C#, jako například atributy, vlastnosti, metody, třídy, event handlery atd. Na konkrétním příkladu níže lze vidět získání dat HTTP požadavkem po načtení stránky, které se provádí v asynchronní metodě OnInitializedAsync, nebo implementaci metod pro obsluhu událostí, jež jsou vyvolány ovládáním tabulky.

Obrázek 13 - Ukázka implementace komponenty

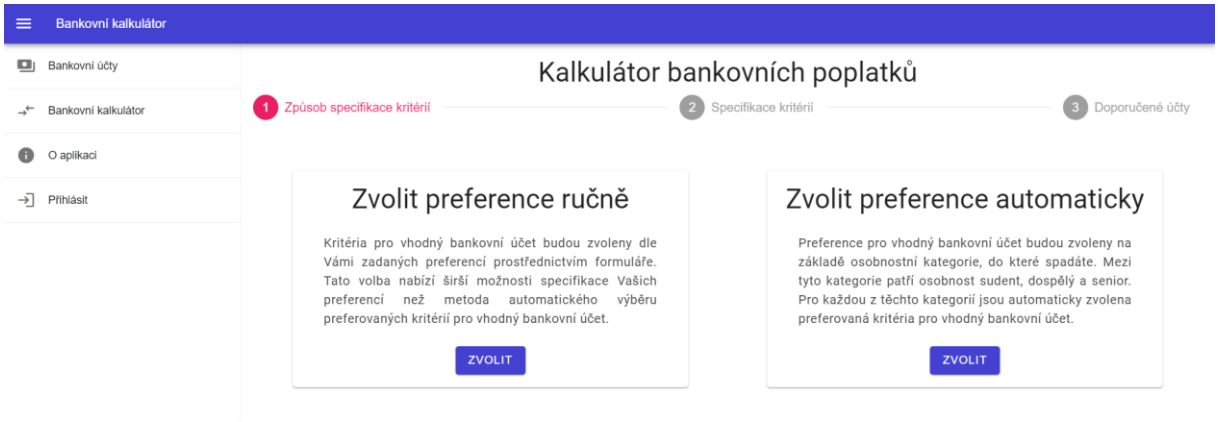
```
@code
{
    RadzenDataGrid<BankAccountDetailDto> dataGrid;
    private List<BankAccountDetailDto> _bankAccounts;

    protected override async Task OnInitializedAsync()
    {
        base.OnInitialized();

        _bankAccounts = await HttpClient.GetFromJsonAsync<List<BankAccountDetailDto>>("BankAccount");
    }

    void RowSelect(BankAccountDetailDto bankAccountDetail)
    {
        Nav.NavigateTo($"/bank-account/{bankAccountDetail.Id}");
    }

    void InsertRow()
    {
        Nav.NavigateTo("/bank-account/create");
    }

    void EditRow(BankAccountDetailDto bankAccount)
    {
        Nav.NavigateTo($"/bank-account/{bankAccount.Id}/edit");
    }

    async Task DeleteRow(BankAccountDetailDto bankAccount)
    {
        if (_bankAccounts.Contains(bankAccount))
        {
            await HttpService.Delete<BankAccountDetailDto>($"BankAccount/{bankAccount.Id}");

            _bankAccounts.Remove(bankAccount);

            await dataGrid.Reload();
        }
    }
}
```

Zdroj: Vlastní zpracování

4.5 Popis funkčnosti aplikace

Po zadání webové adresy aplikace do prohlížeče se zobrazí nejdůležitější okno – a sice okno s možností volby způsobu specifikace kritérií, kterou počíná proces vyhodnocení bankovních účtů. V levé části lze vidět nabídku aplikace, která se mění v závislosti na tom, zda je uživatel přihlášený. Výchozí nabídka pro nepřihlášeného uživatele obsahuje položky „Bankovní účty“, „Bankovní kalkulátor“, „O aplikaci“ a „Přihlásit“. V horní části stránky se nachází komponenta, která zobrazuje informaci o krocích, které je potřeba provést pro úspěšné vyhodnocení bankovních účtů. Na následujícím obrázku lze vidět úvodní nabídku aplikace s prvním krokem pro vyhodnocení bankovních účtů:

Obrázek 14 - Úvodní stránka aplikace



Zdroj: Vlastní zpracování

4.5.1 Analýza bankovních účtů

Výchozí stránka po spuštění aplikace je „Bankovní kalkulátor“. Tato stránka zobrazí uživateli sekvenci kroků, které je pro provedení analýzy nutné provést. Tento proces se skládá ze tří kroků, přičemž se jedná o následující dílčí úkony:

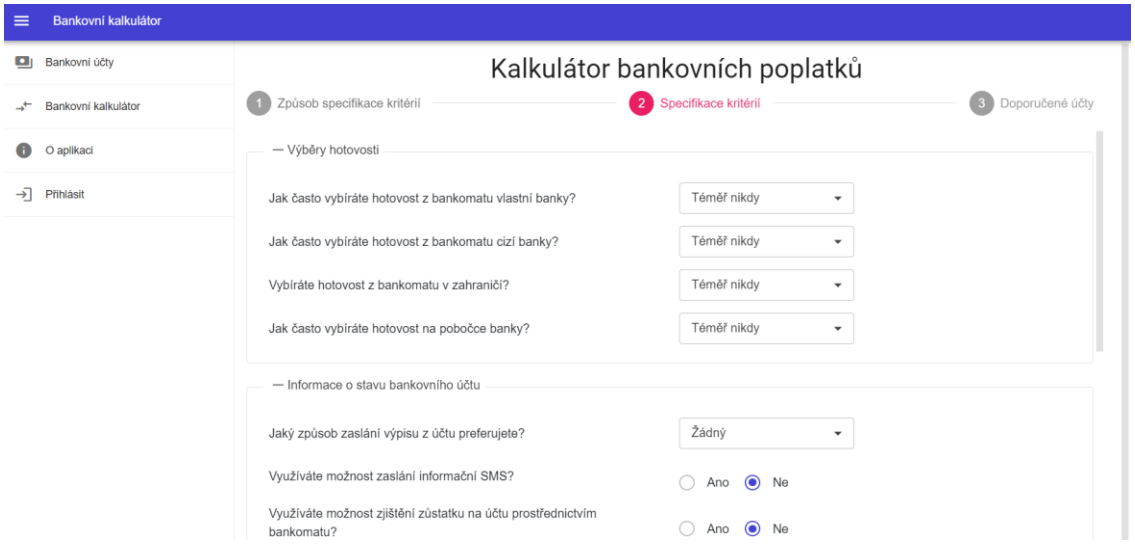
1) způsob specifikace kritérií,
2) specifikace kritérií,
3) doporučené účty.

V prvním kroku je po uživateli vyžadováno zvolení způsobu zadání kritérií, bez něhož nelze dále pokračovat. Aplikace umožňuje zadání preferencí dvěma způsoby: ručně a automaticky. Ruční zadání poskytuje uživateli možnost zadat osobní preference specifičtěji prostřednictvím formuláře, pomocí něhož budou zadány informace například o frekvenci výběrů hotovosti,

počtu odchozích plateb za měsíc, zda je uživatel student atd. Druhá možnost (tedy zvolení preferencí automaticky) je mnohem jednodušší a výrazně urychlí proces analýzy. Zadání preferencí probíhá tím způsobem, že uživatel zvolí kategorii osobnosti, do které spadá, což je jediná vstupní hodnota zadána aplikaci, pomocí níž budou bankovní účty vyhodnoceny. Obě metody zadání preferencí budou popsány později.

Uživatel se přesune na druhý krok vyhodnocení bankovních účtů, jakmile bude zvolena některá z možností pro zadání preferencí bankovního účtu. V případě zvolení možnosti „Zvolit preference ručně“ bude zobrazen tento formulář pro uvedení informací o míře a způsobu využívání bankovního účtu uživatelem:

Obrázek 15 - Formulář pro specifikaci preferencí uživatele

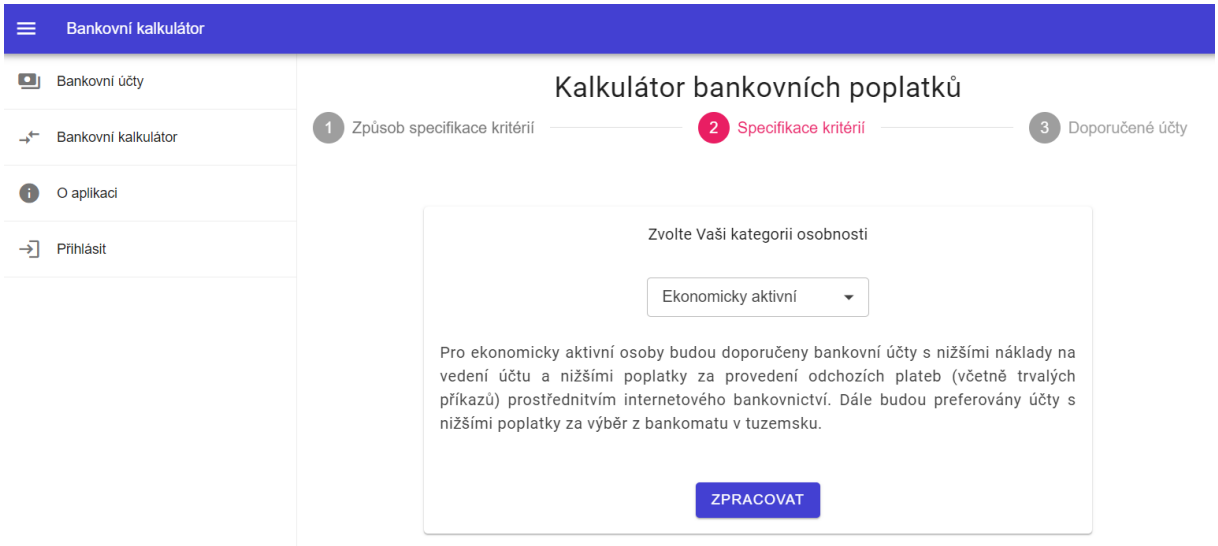


Zdroj: Vlastní zpracování

Formulář nabízí uživateli možnost zadání několika hodnot pro podrobnější analýzu, která je více osobní a individuální. Pomocí tohoto formuláře lze zadat informace spojené s předpokládaným využíváním bankovního účtu (jako například frekvence výběru hotovosti, počet odchozích plateb, způsob zasílání výpisu z účtu atd.) a základní informace o zadavateli, a to pomocí několika odlišných komponent – přepínačů, kombinovaných polí a textových polí. Díky formuláři lze zadat veškeré informace pro určení vah kritérií a následné vyhodnocení bankovních účtů, kromě tří hodnot – a sice hodnotu spjatou s preferovanou hodnotou úrokové míry, měsíčního poplatku za vedení účtu a jednorázového poplatku za založení účtu. Lze totiž předpokládat, že uživatel vyžaduje, aby roční úroková sazba byla, pokud možno co nejvyšší a poplatek za vedení společně s poplatkem za založení účtu naopak co nejnižší. Je tedy zbytečné nabízet uživateli možnost pro určení preferovaných hodnot těchto atributů.

Upřednostňuje-li uživatel jednodušší způsob určení preferencí a zvolí v prvním kroku automatické zadání preferencí, tak bude zobrazen následující obsah stránky:

Obrázek 16 - Specifikace kritérií dle typu osobnosti

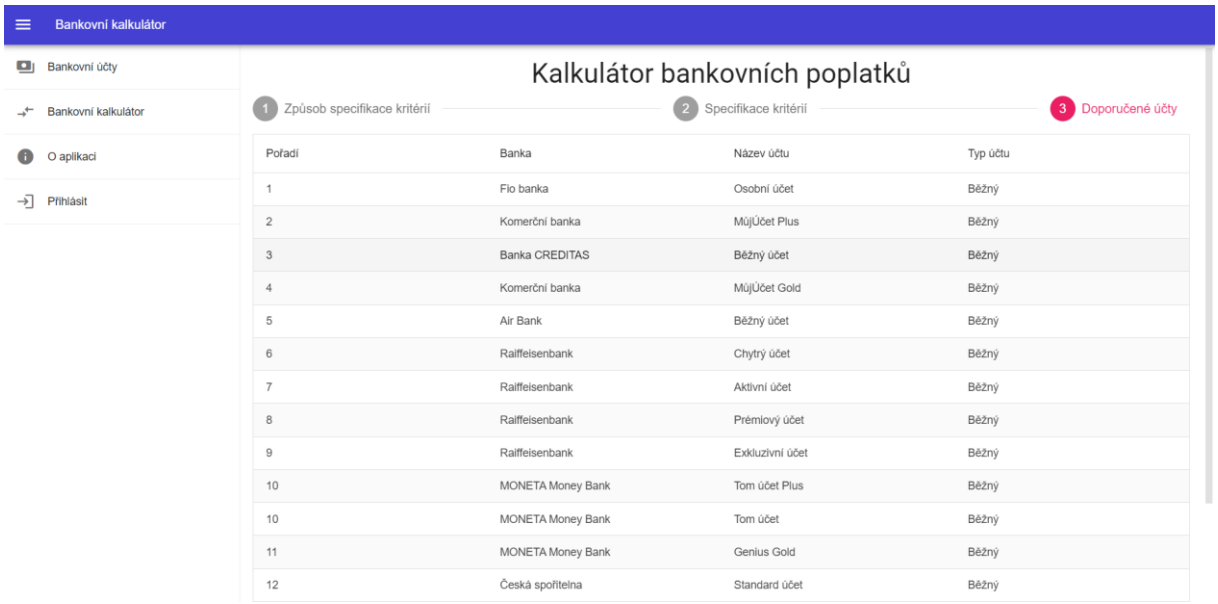


Zdroj: Vlastní zpracování

Uživateli se zobrazí jediná komponenta pro zvolení kategorie osobnosti, přičemž je na výběr z možností: „Ekonomicky aktivní“, „Student“, „Senior“. Po zvolení možnosti z výběru je automaticky generován popisek s podrobným popisem preferovaných vlastností bankovního účtu dle zvolené kategorie. Každá z možností představuje šablonu bodového ohodnocení kritérií. Tyto šablony pracují s určitým předpokladem o zadavateli. Lze například předpokládat, že pokud je zadavatelem osoba v penzi, tak bude preferovat provádění plateb či výběr hotovosti na pobočce, nebo například bude preferovat zaslání výpisu z účtu poštou. U studentů je naopak možné předvídat, že preferují odesílání plateb skrze internetové bankovnictví, chtějí nízký poplatek za výběr z bankomatu a také častěji cestují, s čímž souvisí požadavek nižších poplatků za výběr hotovosti v zahraničí. Navíc pokud je zvolena možnost „Student“, tak do analýzy budou zahrnuty i studentské účty. Pro ekonomicky aktivní osoby je kladen důraz zejména na nízké poplatky za vedení účtu, provádění plateb elektronicky a výběry hotovosti z bankomatů.

Po specifikaci kritérií se uživateli zobrazí stránka, která je určena pro poslední krok pro vyhodnocení bankovních účtů, která zobrazuje tabulku s doporučenými bankovními účty seřazenými dle pořadí. Tabulka kromě toho zobrazuje také název bankovního produktu, název banky, která produkt nabízí, a typ bankovního účtu (zda se jedná o běžný či studentský bankovní účet).

Obrázek 17 - Výsledek analýzy bankovních účtů

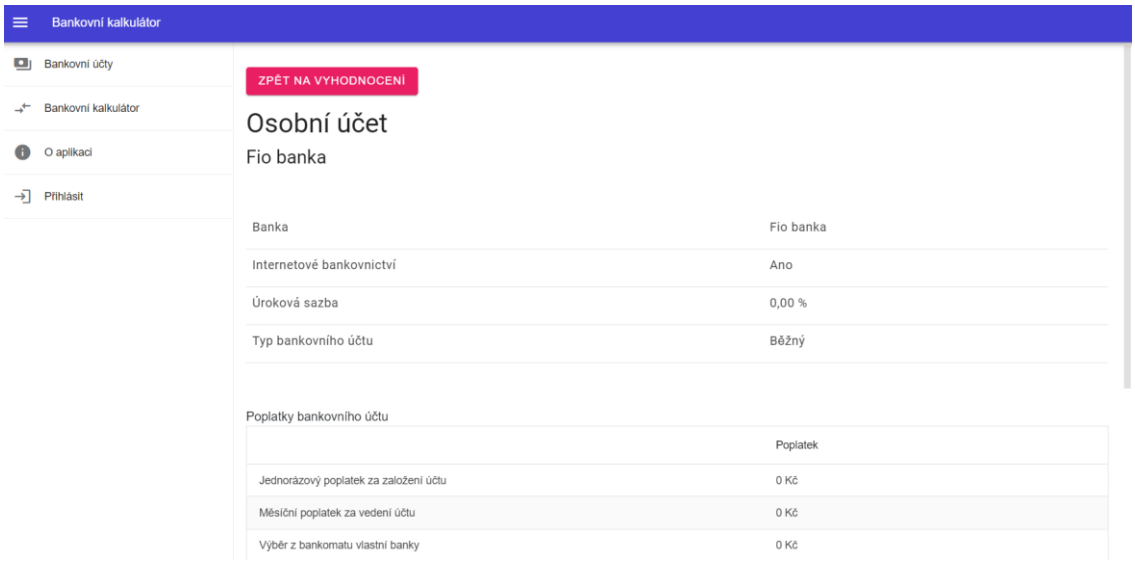


Pořadí	Banka	Název účtu	Typ účtu
1	Fio banka	Osobní účet	Běžný
2	Komerční banka	MůjÚčet Plus	Běžný
3	Banka CREDITAS	Běžný účet	Běžný
4	Komerční banka	MůjÚčet Gold	Běžný
5	Air Bank	Běžný účet	Běžný
6	Raiffeisenbank	Chytrý účet	Běžný
7	Raiffeisenbank	Aktivní účet	Běžný
8	Raiffeisenbank	Prémiový účet	Běžný
9	Raiffeisenbank	Exkluzivní účet	Běžný
10	MONETA Money Bank	Tom účet Plus	Běžný
10	MONETA Money Bank	Tom účet	Běžný
11	MONETA Money Bank	Genius Gold	Běžný
12	Česká spořitelna	Standard účet	Běžný

Zdroj: Vlastní zpracování

Po kliknutí na bankovní účet ze seznamu se uživateli zobrazí stránka s detailem účtu, která poskytuje uživateli podrobné informace včetně obecných informací o bankovním účtu (název banky a bankovního účtu, úroková sazba, typ účtu) a poplatky spojené s jeho využíváním, které jsou uvedeny v korunách. Kliknutím na tlačítko „Zpět na vyhodnocení“ v horní části stránky se uživatel vrátí na zobrazení tabulky s vyhodnocením bankovních účtů.

Obrázek 18 - Detail bankovního účtu

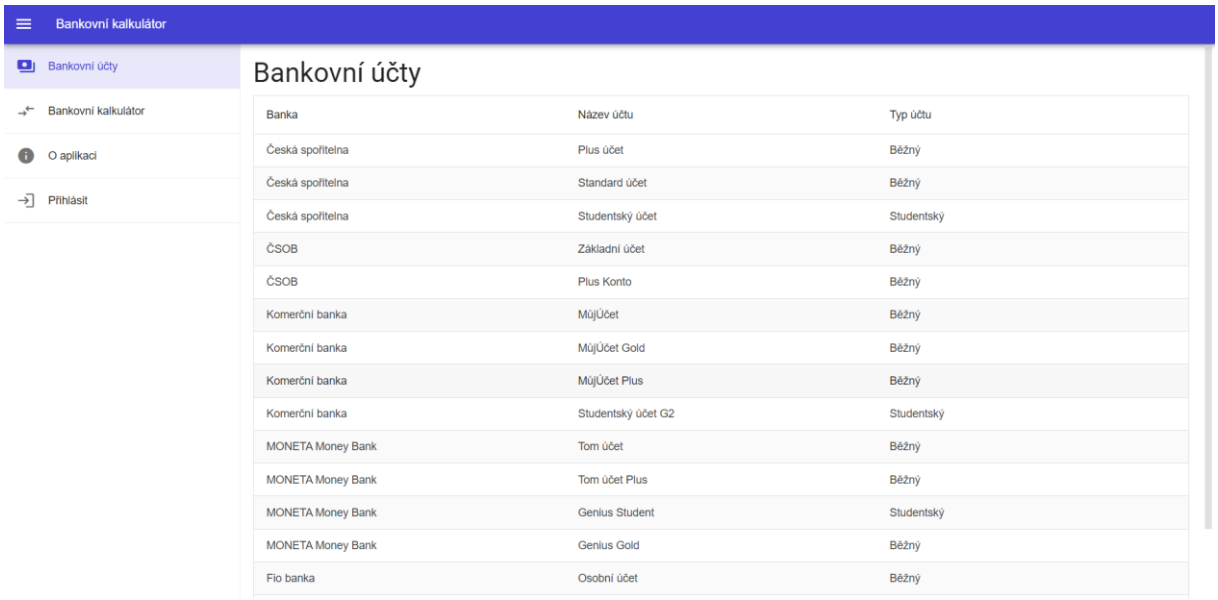


Zdroj: Vlastní zpracování

4.5.2 Zobrazení seznamu bankovních účtů

Kliknutím na možnost „Bankovní účty“ z nabídky se uživateli zobrazí seznam všech bankovních účtů, jejichž informace jsou v aplikaci uloženy a které jsou vyhodnocovány pomocí metod vícekriteriálního rozhodování. Tabulka je velmi podobná tabulce pro zobrazení vyhodnocení bankovních účtů – neposkytuje však informaci o konečném pořadí bankovního účtu, jelikož se jedná o informaci získanou na základě analýzy variant. Kliknutím na libovolnou položku tabulky se opět zobrazí detail konkrétního bankovního účtu. Záznamy lze stejně jako v případě tabulky s informacemi o vyhodnocení bankovních účtů seřadit dle názvu banky, názvu nebo typu účtu.

Obrázek 19 - Přehled bankovních účtů



Zdroj: Vlastní zpracování

4.5.3 Informace o aplikaci

Kliknutím na poslední dostupnou položku z nabídky aplikace „O aplikaci“ se uživateli zobrazí obecné informace o aplikaci. Uživatele informuje o účelu aplikace, stručně shrnuje dostupné informace o bankovních účtech a poskytuje stručný úvod do problematiky vícekriteriálního rozhodování.

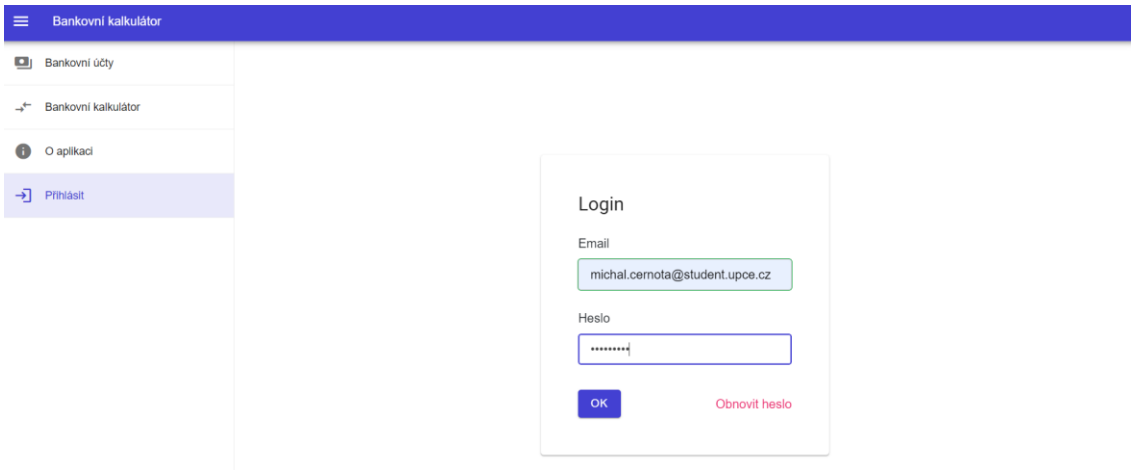
Obrázek 20 - Informace o aplikaci

Zdroj: Vlastní zpracování

4.5.4 Přihlášení uživatele

Uživatel se může přihlásit vyplněním autentizačních údajů (e-mailu a hesla) do následujícího formuláře:

Obrázek 21 - Formulář pro přihlášení uživatele



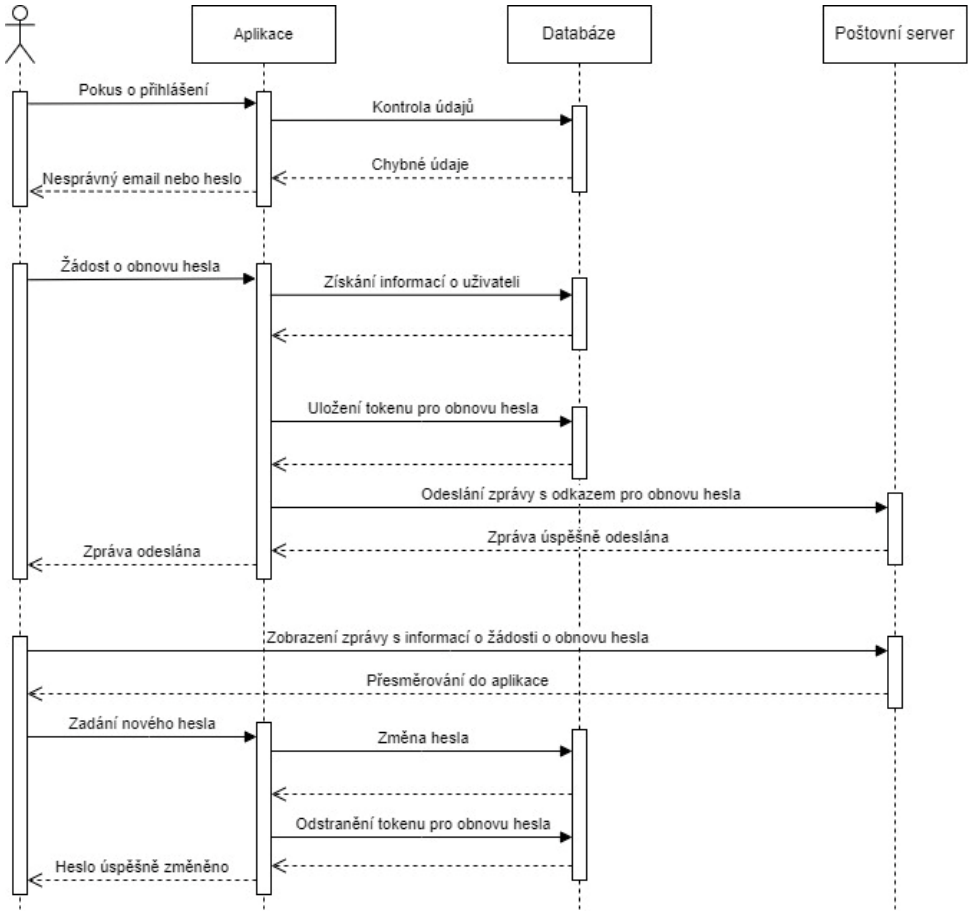
Zdroj: Vlastní zpracování

V případě úspěšného přihlášení je uživatel autorizován po dobu platnosti JWT tokenu, který je uložen v uložišti prohlížeče na straně uživatele. Po vypršení jeho platnosti je uživatel na tuto skutečnost upozorněn, dojde k odhlášení uživatele, a je nutné zopakovat proces přihlášení. Po

úspěšném přihlášení se v nabídce zobrazí mimo jiné položka „Odhlásit“. Po kliknutí na tuto položku dojde k odhlášení uživatele, odstranění JWT tokenu z uložiště prohlížeče, a zobrazení původní nabídky pro nepřihlášeného uživatele. Další položka nabídky, která přibude po úspěšné autentizaci uživatele, je položka „Banky“, která zobrazí stránku s přehledem bankovních subjektů, jejichž informace smí autorizovaný uživatel editovat. Nepřihlášenému uživateli tato položka není zobrazeno proto, že aplikace je cílena na vyhodnocení bankovních účtů, a proto informace o bankovních subjektech nejsou pro běžného uživatele podstatné.

Aplikace taktéž umožňuje obnovení zapomenutého hesla. V případě žádosti o obnovu hesla je uživateli na emailovou adresu prostřednictvím SMTP protokolu zaslána elektronická zpráva, která obsahuje URL odkaz, který uživatele přesměruje na stránku aplikace s formulářem pro obnovu hesla. Tento URL odkaz obsahuje mimo jiné token pro obnovu hesla, jehož platnost je jeden den, a který slouží k identifikaci uživatelského účtu, jehož heslo bude obnoveno. Proces obnovy hesla znázorňuje následující sekvenční diagram:

Obrázek 22 - Proces obnovy hesla

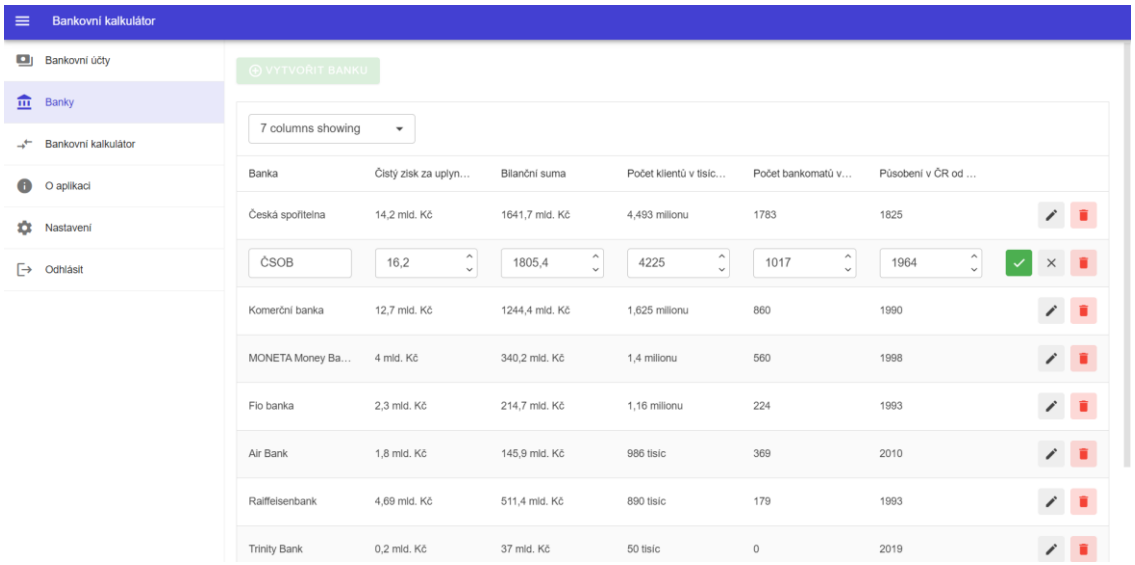


Zdroj: Vlastní zpracování

4.5.5 Vytváření, editace, mazání dat

Aplikace umožnuje autorizovanému uživateli vytvářet, editovat a odstraňovat data. Pracuje-li uživatel s daty vztahujícími se k datové entitě banky, pak je možné vytvářet a editovat data v rámci takzvaného in-line režimu, kdy se atributy banky zadávají přímo v tabulce zobrazující informace o bankách. Proces editace lze potvrdit či stornovat kliknutím na tlačítko nacházející se na pravé straně tabulky.

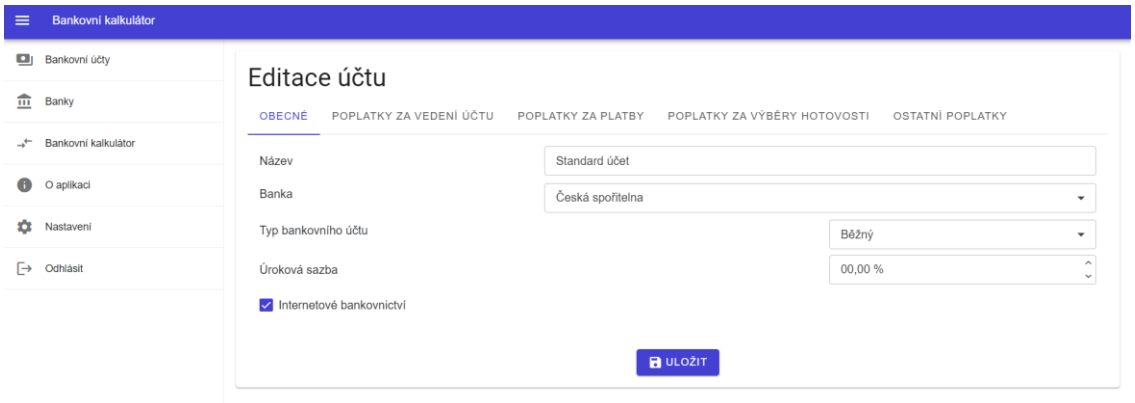
Obrázek 23 - In-line editace banky



Zdroj: Vlastní zpracování

Pro vytváření bankovního účtu již bylo potřeba vytvořit formulář pro určení hodnot atributů bankovního účtu, jelikož na rozdíl od banky disponuje bankovní účet větším počtem informací a in-line editace by tak již nebyla možná. Pro tyto účely byl vytvořen následující vícestránkový formulář:

Obrázek 24 - Editace bankovního účtu



Zdroj: Vlastní zpracování

4.5.6 Nastavení aplikace

Autorizovaný uživatel může také modifikovat nastavení aplikace či registrovat nový uživatelský účet. Jediné nastavení aplikace, které přihlášený uživatel může v současnosti modifikovat, je metoda vícekriteriálního rozhodování, která bude použita pro analýzu bankovních účtů.

Obrázek 25 - Nastavení aplikace

Zdroj: Vlastní zpracování

4.6 Způsob vyhodnocení variant

V následující kapitole bude popsán průběh zpracování požadavku na vyhodnocení bankovních účtů, včetně použitých metod pro stanovení vah kritérií a vyhodnocení variant.

4.6.1 Bodové ohodnocení kritérií v klientské aplikaci

Klientská aplikace po zadání preferencí uživatele určuje bodová ohodnocení pro jednotlivá kritéria. Nejedná se však o váhy – ty jsou vypočítány v rámci zpracování požadavku aplikačním rozhraním aplikace. Hodnoty pro tato ohodnocení spadají do intervalu <1;10>. Po ohodnocení kritérií je odeslán požadavek na aplikační rozhraní pro zpracování a určení nejvhodnějších bankovních účtů.

4.6.2 Zpracování požadavku na vyhodnocení bankovních účtů

Zpracování požadavku pro vyhodnocení bankovních účtů lze popsat v několika krocích. Obecně proces vyhodnocení obnáší získání dat z úložiště, konverzi dat na varianty, určení vah kritérií, sestavení a zpracování matice variant metodou vícekriteriálního rozhodování a odeslání odpovědi zpět na klienta.

Nejprve je potřeba získat všechna kritéria pro bankovní účty, včetně jejich popisu a informací o typu kritérií (zda se jedná o minimalizační či maximalizační kritéria). Kritériím bankovních účtů se následně přiřadí váhy dle bodového ohodnocení, které bylo zadáno uživatelem

v aplikaci webového klienta, a to pomocí jedné z implementovaných metod pro určení vah kritérií – tedy prostřednictvím bodovací metody nebo pomocí Saatyho metody. Výchozí metoda pro stanovení vah kritérií je bodovací metoda.

Poté je určen index spolehlivosti banky. Jedná se o hodnotu, která určuje míru spolehlivosti banky na základě informací o bankovním subjektu, které vypovídají zejména o jeho míře upřednostnění klienty v České republice. Jde o data dostupná z výročních a tiskových zpráv a z povinně zveřejňovaných informací bank. Údaje, na jejichž základě je určen index spolehlivosti banky, jsou:

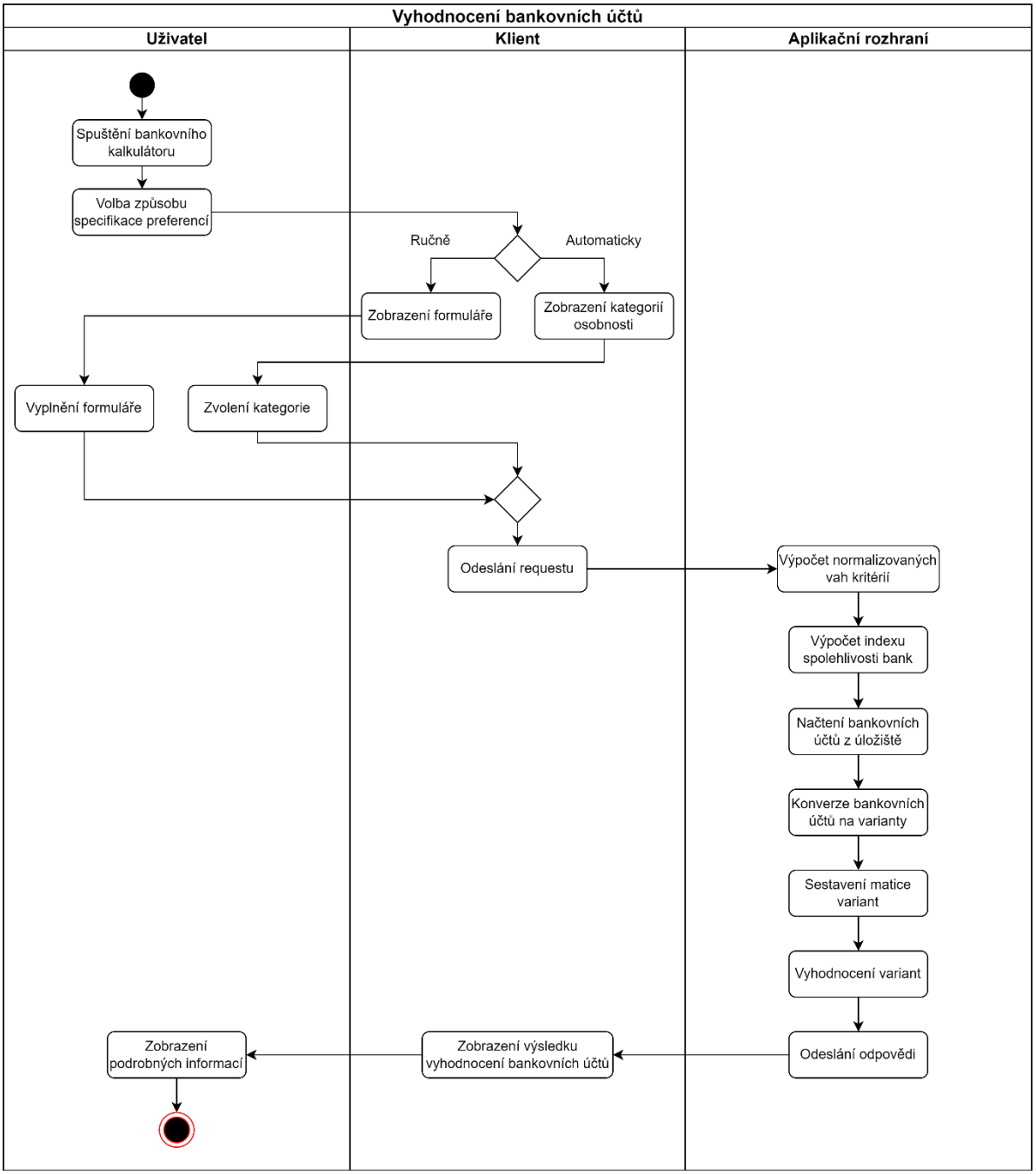
- bilanční suma,
- počet klientů,
- čistý zisk za roční období,
- rok začátku působení banky na území České republiky,
- počet bankomatů v České republice.

Index spolehlivosti bank je určen pomocí metody váženého součtu. Váhy zmíněných kritérií indexu spolehlivosti banky jsou stanoveny bodovací metodou, avšak váhy je možné určit i pomocí konstant, jelikož tyto váhy nejsou explicitně zadávány uživatelem. Každému kritériu je v tomto případě přiřazeno stejné ohodnocení a každé kritérium pro určení indexu spolehlivosti banky má tak stejnou váhu. Hodnota užitku pak představuje informaci vyjadřující spolehlivost banky.

Po získání indexu spolehlivosti banky se získají potřebná data pro analýzu bankovních účtů z databáze. Entity bankovních účtů jsou následně konvertovány na varianty, kterým jsou přiřazeny odpovídající kritéria s jejich hodnotami. Poté je sestrojena kriteriální matice, která je následně zpracována pomocí určené metody pro vícekriteriální rozhodování, jenž je specifikována v nastavení aplikace. Varianty (v tomto případě bankovní účty s informací o konečném pořadí) jsou nakonec odeslány zpět klientské aplikaci, která je ve vhodné formě zobrazí uživateli.

Proces, jehož výsledkem je vyhodnocení bankovních účtů je s největší pravděpodobností nejkomplexnější a nejdůležitější business proces aplikace, a proto je vhodné tento proces také popsat pomocí diagramu aktivit, který znázorňuje řídící tok a sekvenci provedených kroků, které vedou k dosažení žádaného výsledku. Diagram, který lze nalézt na následující straně, detailněji popisuje proces vyhodnocení bankovních účtů:

Obrázek 26 - Proces vyhodnocení bankovních účtů



Zdroj: Vlastní zpracování

5 ANALÝZA BANKOVNÍCH ÚČTŮ

V této kapitole bude popsána použitá metodika pro sběr dat a jejich analýzu, včetně výsledků analýzy bankovních účtů. Dále bude zdůvodněno rozhodnutí pro zvolení konkrétních bankovních účtů (respektive bankovních subjektů) pro analýzu.

5.1 Popis dat

Byly analyzovány bankovní produkty deseti největších bank na základě bilanční sumy (objem spravovaných peněz), kterou disponovaly v kalendářním roce 2021, a to z důvodu zredukování počtu analyzovaných bankovních produktů. Informace o bankovních subjektech byly získány z výročních zpráv za uvedené období. V původním datovém vzorku byly rovněž zahrnuty banky Sberbank a Equa bank, avšak tyto bankovní subjekty musely být vyjmuty, jelikož Sberbank je v současnosti v procesu odebírání bankovní licence a Equa bank se koncem roku 2022 sloučila s Raiffeisenbank, která Equa bank nahradila, včetně všech jejích nabízených bankovních produktů a služeb.

Data bankovních subjektů, která budou během analýzy použita pro výpočet indexu spolehlivosti banky, tj. jednoho z vybraných kritérií variant, respektive bankovních účtů, byla získána z výročních zpráv vztahujících se k roku 2021. Pro výpočet jmenovaného indexu, který představuje jediné použité kvalitativní kritérium, byly zjištěny hodnoty bilanční sumy, čistého zisku, počtu klientů, počtu bankomatů v ČR a rok, od kterého začala banka působit na tuzemském trhu. Získané informace lze vidět v následující tabulce:

Tabulka 3 - Bankovní subjekty

	Bilanční suma (mld. Kč)	**Počet klientů (tis.)**	**Čistý zisk (mld. Kč)**	**Působení od roku**	**Počet bankomatů**
ČSOB	1805,4	4225	16,2	1964	1017
Česká spořitelna	1641,7	4493	14,2	1825	1783
Komerční banka	1244,4	1625	12,7	1990	860
UniCredit Bank	693,4	450	6,98	1998	264
Raiffeisenbank	511,4	890	4,69	1993	179
Moneta	340,2	1400	4	1998	560
Fio banka	214,7	1160	2,3	1993	224
Air Bank	145,9	986	1,8	2010	369
Banka CREDITAS	62,9	130	0,1143	1996	0
Trinity Bank	37	50	0,2	2019	0

Zdroj: [30], [31], [32], [33], [34], [35], [36], [37], [38], [39]

Všechny z výše uvedených informací představují maximalizační kritéria, s výjimkou časové informace o počátku působení banky. Index spolehlivosti banky určuje numerickou hodnotu, která určuje významnost tuzemské banky. Klient, který se rozhoduje nad možnostmi výběru bankovního účtu, totiž může preferovat účet, který nabízí banka například s vyšším počtem klientů, která je více oblíbená, disponuje vyšší bilanční sumou (či ziskem za roční období) a která na bankovním trhu působí delší dobu.

Dále bylo zapotřebí získat informace o bankovních produktech zvolených bank. Bylo rozhodnuto, že budou analyzovány jak běžné bankovní účty, tak studentské, a to z důvodu vyšší míry využití bankovního kalkulátoru. Proces získání informací o bankovních účtech se tak týkal standardních bankovních produktů pro fyzické osoby. Byla zvolena nejběžnější kritéria bankovních účtů, které se týkají víceméně každého uživatele, a tak bude aplikace uživateli poskytovat co nejpřesnější výsledek analýzy. Konečný datový soubor bankovních účtů tak obsahuje 23 atributů (z nichž 19 je numerických) a 26 entit (bankovních účtů, z nichž je pět studentských). Datový soubor se skládá z následujících atributů, na jejichž základě proběhne analýza variant:

- název bankovního účtu,
- název banky,
- typ bankovního účtu – studentský nebo běžný,
- internetové bankovnictví – informace, zda k bankovnímu účtu lze zřídit internetové bankovnictví,
- úroková sazba p. a. – informace o procentním zvýšení uložené částky za roční období,
- jednorázový poplatek za založení účtu,
- měsíční poplatek za vedení účtu,
- poplatek za výběr hotovosti z bankomatu v ČR,
- poplatek za výběr hotovosti z bankomatu jiné banky v ČR,
- poplatek za výběr hotovosti z bankomatu v zahraničí,
- poplatek za výběr hotovosti z bankomatu cizí banky v zahraničí,
- poplatek za dotaz na zůstatek prostřednictvím bankomatu,
- poplatek za dotaz na zůstatek prostřednictvím bankomatu cizí banky,

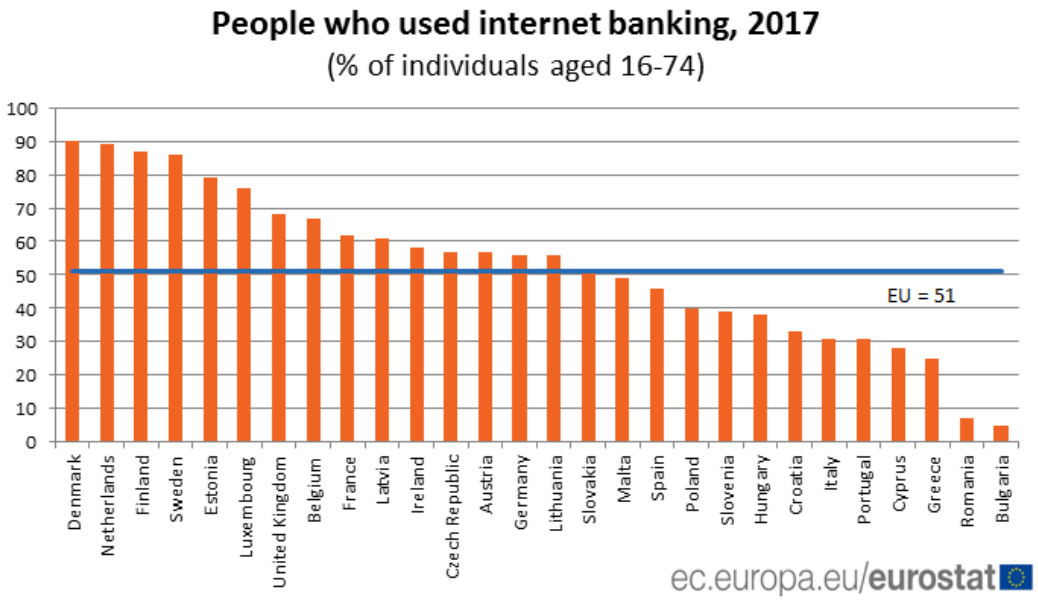
- poplatek za výběr hotovosti na pobočce banky,
- poplatek za výpis z účtu zaslaný elektronicky,
- poplatek za výpis z účtu zaslaný poštou,
- poplatek za zaslání informační SMS,
- poplatek za jednorázovou platbu přes internet,
- poplatek za jednorázovou platbu na pobočce,
- poplatek za zřízení/provedení trvalého příkazu na pobočce,
- poplatek za zřízení/provedení trvalého příkazu přes internet.

Z výčtu kritérií lze vidět, že byla zvolena taková kritéria, která odrážejí možnosti současné doby (například využívání internetového bankovnictví, možnost platby přes internet), ale i v dnešní době již nepříliš využívaných možností využívání bankovních účtů, jako například možnost zadání platby na pobočce banky. Volba různorodých kritérií byla učiněna z důvodu potencionálnímu využití bankovního kalkulátoru různými věkovými generacemi, které mají odlišné preference.

Z výše uvedeného výčtu atributů lze také vypozorovat, že se ve většině případů jedná o informaci o určitém poplatku za využívání bankovního účtu. Většina zvolených kritérií je tak minimalizačního typu.

Jak lze vidět na obrázku níže, Česká republika patří mezi země s největší mírou využívání internetového bankovnictví z Evropských zemí. Lze tak předpokládat, že zahrnutí kritérií souvisejících s internetovým bankovnictvím je jedním z důležitých požadavků analýzy bankovních účtů.

Obrázek 27 - Míra využití internetového bankovnictví v EU



Zdroj: [15]

Naopak ačkoli některé banky již ruší možnost platby na pobočce, někteří uživatelé bankovních účtů stále preferují tento způsob provádění bankovních transakcí, a to zejména lidé v penzi.

Informace vztahující se k bankovním účtům jsou aktuální k datu 18. 2. 2023. Jde o podstatnou informaci, jelikož banky průběžně upravují sazebníky, což má za následek postupnou neaktuálnost dat. Z toho důvodu bylo během zpracování této práce zapotřebí několikrát aktualizovat data dle právě stanovených sazeb bankovních účtů.

5.2 Metodika analýzy bankovních účtů

Jak již bylo zmíněno v kapitole „4.6 Způsob vyhodnocení variant“, tak analýzu bankovních účtů lze pomocí aplikace provádět několika metodami vícekriteriálního rozhodování, ale i váhy kritérií lze také stanovit více metodami. V této kapitole bude popsána analýza bankovních účtů pomocí každé implementované metody vícekriteriálního rozhodování, přičemž budou prezentovány výsledky analýzy bankovních účtů, které byly získány použitím těchto metod.

Aby bylo možné výsledky analýzy bankovních účtů mezi sebou vzájemně porovnat, bylo nutné určit několik scénářů, které budou použity pro každou analýzu, čímž vzniknou stejné podmínky pro aplikaci metod vícekriteriálního rozhodování. Zejména je potřeba použít identické váhy kritérií pro vyhodnocení variant každou metodou vícekriteriálního rozhodování. Vzhledem k tomu, že praktická část práce poskytuje dvě možnosti stanovení vah kritérií (automatické

zvolení vah kritérií dle kategorie osobnosti a manuální stanovení vah vyplněním formuláře), tak bylo zvoleno automatické stanovení vah kritérií z důvodu omezení chybovosti při zadávání údajů pro analýzu bankovních účtů, jelikož při vyplňování formuláře dojde k chybě s vyšší pravděpodobností než v případě zadání jediné vstupní hodnoty. Bude tak provedena analýza bankovních účtů pro tři scénáře – tedy vyhodnocení bankovních účtů pro studenty, dospělé osoby a osoby v penzi.

Bankovní účty byly analyzovány téměř všemi implementovanými metodami vícekriteriálního rozhodování. Pro analýzu nelze využít metodu bazální varianty z důvodu nekompatibilních dat (důsledkem výskytu poplatků s nulovou hodnotou docházelo k sestavení nevalidní matice variant a chybným matematickým operacím). Proto bude provedena analýza bankovních účtů metodou váženého součtu, metodou TOPSIS, Lexikografickou metodou a metodou pořadí, a to pro všechny zmiňované modely – tedy modely „Student“, „Ekonomicky aktivní“ a „Senior“.

5.2.1 Stanovení vah kritérií

Váhy kritérií byly stanoveny pomocí bodovací metody. Jedná se o preferovanější metodu oproti Saatyho metodě z důvodu předpokladů pro použití této metody. Saatyho metoda je totiž založena na párovém srovnávání kritérií. Ačkoli aplikace umožňuje párově srovnávat kritéria jednoduchou modifikací procesu stanovení vah kritérií, kdy míra preference mezi kritérii je určena na základě bodového ohodnocení obou kritérií na předem určené stupnici, tak metoda bodovací je v tomto případě preferována, jelikož určení vah kritérií vychází z bodového ohodnocení kritérií na základě stanovených preferencí uživatele a následné normalizaci vah, což jsou vhodné podmínky pro použití této metody. Z tohoto důvodu je pro aplikaci preferována právě tato metoda. Tabulka na následující straně demonstruje použité normalizované váhy kritérií:

Tabulka 4 - Váhy kritérií pro hodnocení bankovních účtů

Kritérium	Model ekonomicky aktivní	Model student	Model senior
Index spolehlivosti banky	0,07767	0,07447	0,04630
Internetové bankovnictví	0,08738	0,10638	0,04630
Úroková míra	0,02913	0,02128	0,02778
Poplatek za založení účtu	0,02913	0,03191	0,05556
Poplatek za vedení účtu	0,04854	0,05319	0,07407
Poplatek za výběr z bankomatu	0,04854	0,04255	0,06481
Poplatek za výběr z bankomatu v zahraničí	0,06796	0,08511	0,04630
Poplatek za výběr z bankomatu cizí banky	0,03883	0,03191	0,05556
Poplatek za výběr z bankomatu cizí banky v zahraničí	0,05825	0,07447	0,03704
Poplatek za výběr na pobočce	0,02913	0,02128	0,06481
Dotaz na zůstatek na účtu z bankomatu	0,03883	0,03191	0,04630
Dotaz na zůstatek na účtu z bankomatu cizí banky	0,02913	0,02128	0,03704
Poplatek za informační SMS	0,01942	0,02128	0,02778
Poplatek za výpis z účtu elektronicky	0,04854	0,05319	0,02778
Poplatek za výpis z účtu poštou	0,05825	0,03191	0,05556
Poplatek za odchozí platbu odeslanou přes internet	0,07767	0,08511	0,05556
Poplatek za odchozí platbu odeslanou z pobočky	0,01942	0,02128	0,04630
Poplatek za zřízení trvalé platby přes internet	0,05825	0,06383	0,03704
Poplatek za provedení trvalé platby přes internet	0,07767	0,08511	0,05556
Poplatek za zřízení trvalé platby na pobočce	0,01942	0,01064	0,03704
Poplatek za provedení trvalé platby na pobočce	0,03883	0,03191	0,05556
Σ	1	1	1

Zdroj: Vlastní zpracování

5.3 Vlastní výsledky šetření

V této kapitole budou popsány výsledky analýzy bankovních účtů aplikováním implementovaných metod vícekriteriálního rozhodování a stanovených vah kritérií pro zvolené modely. Výsledky analýzy studentských bankovních účtů nejsou pro analyzované modely „Ekonomicky aktivní“ a „Senior“ dostupné. Není totiž žádoucí, aby uživatelé, kteří do těchto kategorií spadají, tento typ bankovního účtu využívali. Také je potřeba zmínit skutečnost, že jelikož bylo zvoleno omezené množství atributů bankovních účtů pro analýzu, tak některé bankovní účty z množiny (zejména ty, které jsou poskytovány totožným bankovním subjektem) disponují identickými hodnotami poplatků, a tak je jejich výsledné hodnocení totožné.

5.3.1 Metoda pořadí

Ačkoli metoda pořadí nevyžaduje informaci o preferenci kritérií, tak i tato metoda byla použita pro analýzu bankovních účtů. Umožňuje totiž mimo jiné identifikovat ideální bankovní účet z množiny neboli určit ty bankovní účty, které disponují nejnižšími poplatky z dostupných dat.

Tabulka 5 - Vyhodnocení bankovních účtů metodou pořadí

Banka	**Účet**	**Ekonomicky aktivní**		**Student**		**Senior**	
		Pořadí	**Hodnocení**	**Pořadí**	**Hodnocení**	**Pořadí**	**Hodnocení**
Air Bank	Běžný účet	2	108,5	3	139	2	108,5
	Studentský účet			3	139		
Banka CREDITAS	Běžný účet	3	123,5	4	159	3	123,5
Česká spořitelna	Studentský účet			7	182		
	Standard účet	6	150	10	195	6	150
	Plus účet	11	160	14	207,5	11	160
ČSOB	Plus Konto	16	186	19	240,5	16	186
	Základní účet	17	211	20	273	17	211
Fio banka	Účet pro studenty			**1**	**121**		
	Osobní účet	**1**	**99,5**	2	127,5	**1**	**99,5**
Komerční banka	MůjÚčet Plus	4	132,5	5	173,5	4	132,5
	MůjÚčet Gold	5	136,5	6	177,5	5	136,5
	Studentský účet G2			8	189,5		
	MůjÚčet	15	179,5	18	230,5	15	179,5
MONETA Money Bank	Tom účet	8	152	11	196	8	152
	Tom účet Plus	8	152	11	196	8	152
	Genius Student			11	196		
	Genius Gold	9	155,5	12	198,5	9	155,5
Raiffeisenbank	Chytrý účet	12	164,5	15	210	12	164,5
	Aktivní účet	14	172,5	17	220,5	14	172,5
	Prémiový účet	13	166	16	210,5	13	166
	Exkluzivní účet	7	150,5	9	190	7	150,5
Trinity Bank	Běžný účet	10	159	13	203,5	10	159
UniCredit Bank	Účet START	18	228	21	287,5	18	228
	Účet OPEN	19	237	22	299	19	237
	Účet TOP	20	241	23	303	20	241

Zdroj: Vlastní zpracování

Na základě výsledků vyhodnocení variant lze zpozorovat, že v případě analyzovaného modelu „Ekonomicky aktivní“ a „Senior“ je shodně určen Osobní účet od Fio banky jako nejvýhodnější varianta. V případě modelu student byl za optimální variantu označen studentský účet taktéž od Fio banky, jelikož oproti běžnému účtu od identické banky poskytuje bezplatný výběr z bankomatu ostatních bank v zahraničí.

Metodu pořadí však nelze doporučit pro analýzu bankovních účtů prostřednictvím implementované aplikace, jelikož tato metoda nevyžaduje informaci o preferenci kritérií. Nelze tak pracovat se vstupními informacemi pro analýzu, které byly uživatelem specifikovány, což vede k neobjektivním výsledkům analýzy.

5.3.2 Lexikografická metoda

Další použitou metodou byla Lexikografická metoda, která pracuje s ordinální informací o kritériích (tzn. největší vliv na výběr kompromisní varianty má nejdůležitější kritérium), a tak na rozdíl od již popsané metody pořadí je výsledek ovlivněn preferencemi, které byly uživatelem specifikovány. Jelikož jediným výstupem této metody je pořadí variant, tak na rozdíl od ostatních využitých metod analýzy variant, není dostupná žádná hodnota, která by popisovala konečné pořadí varianty.

Tabulka 6 - Vyhodnocení bankovních účtů Lexikografickou metodou

Banka	**Účet**	**Ekonomicky aktivní**	**Student**	**Senior**
		Pořadí	**Pořadí**	**Pořadí**
Air Bank	Běžný účet	18	13	2
	Studentský účet		13	
Banka CREDITAS	Běžný účet	19	14	3
Česká spořitelna	Studentský účet		**1**	
	Standard účet	**1**	16	7
	Plus účet	2	17	15
ČSOB	Plus Konto	3	19	6
	Základní účet	4	20	11
Fio banka	Účet pro studenty		11	
	Osobní účet	17	12	**1**
Komerční banka	MůjÚčet Plus	5	2	12
	MůjÚčet Gold	6	3	16
	Studentský účet G2		4	
	MůjÚčet	7	15	5
MONETA Money Bank	Tom účet	11	5	4
	Tom účet Plus	11	5	4
	Genius Student		5	
	Genius Gold	12	6	19
Raiffeisenbank	Chytrý účet	13	7	8
	Aktivní účet	14	8	13
	Prémiový účet	15	9	17
	Exkluzivní účet	16	10	20
Trinity Bank	Běžný účet	20	18	9
UniCredit Bank	Účet START	8	21	10
	Účet OPEN	9	22	14
	Účet TOP	10	23	18

Zdroj: Vlastní zpracování

Z výsledků výše je patrné, že nejvýhodnější účet pro dospělé a studenty je běžný a studentský účet od České spořitelny. Pro model „Senior“ pak byl jako nejvýhodnější účet určen běžný účet od Fio banky, a to zejména z důvodu nízkých poplatků za provádění bankovních operací na pobočce banky.

Jelikož Lexikografická metoda funguje na principu postupné selekce vyhovující varianty dle nejdůležitějšího kritéria a vyžaduje pouze jednoduchou ordinální informaci o kritériích, tak stejně jako v případě metody pořadí není použití této metody doporučeno pro analýzu bankovních účtů implementovanou aplikací a výsledky této metody jsou spíše orientační. Některá kritéria navíc disponují identickou váhou, což vede také ke zkreslení výsledků analýzy variant, jelikož v tomto případě nelze na základě stanovených vah určit, které kritérium bude upřednostněno při provádění analýzy Lexikografickou metodou.

5.3.3 Metoda TOPSIS

Následující tabulka ukazuje výsledné pořadí variant a jejich relativní vzdálenost od bazální varianty, které byly získány aplikováním metody TOPSIS. Tato metoda vychází z předpokladu, že nejlepší varianta je ta, která disponuje nejkratší vzdáleností od ideální varianty a tím pádem největší vzdáleností od varianty bazální.

Tabulka 7 - Vyhodnocení bankovních účtů metodou TOPSIS

Banka	Účet	Ekonomicky aktivní		Student		Senior	
		Pořadí	Hodnocení	Pořadí	Hodnocení	Pořadí	Hodnocení
Air Bank	Běžný účet	10	0,66680	11	0,70099	3	0,75743
	Studentský účet			11	0,70099		
Banka CREDITAS	Běžný účet	11	0,64208	12	0,69561	10	0,67760
Česká spořitelna	Studentský účet			3	0,76991		
	Standard účet	7	0,68918	14	0,66716	8	0,70864
	Plus účet	9	0,68112	15	0,65757	9	0,69116
ČSOB	Plus Konto	14	0,57850	18	0,56125	13	0,62452
	Základní účet	17	0,44981	20	0,43005	16	0,51612
Fio banka	Účet pro studenty			4	0,76175		
	Osobní účet	6	0,69954	9	0,73252	**1**	**0,80804**
Komerční banka	MůjÚčet Plus	**1**	**0,78486**	**1**	**0,82625**	2	0,78259
	MůjÚčet Gold	2	0,76767	2	0,80339	4	0,74761
	Studentský účet G2			8	0,73372		
	MůjÚčet	16	0,47350	19	0,47716	15	0,57103
MONETA Money Bank	Tom účet	5	0,70017	5	0,76049	5	0,74072
	Tom účet Plus	5	0,70017	5	0,76049	5	0,74072
	Genius Student			5	0,76049		
	Genius Gold	12	0,63515	13	0,67062	14	0,60090
Raiffeisenbank	Chytrý účet	3	0,70862	6	0,75468	6	0,72518
	Aktivní účet	4	0,70510	7	0,75094	7	0,71821
	Prémiový účet	8	0,68291	10	0,71996	11	0,67407
	Exkluzivní účet	13	0,59022	16	0,60115	17	0,50817
Trinity Bank	Běžný účet	15	0,55372	17	0,57220	12	0,62820
UniCredit Bank	Účet START	18	0,35446	21	0,36575	18	0,47115
	Účet OPEN	19	0,32967	22	0,33915	19	0,44101
	Účet TOP	20	0,26361	23	0,26623	20	0,34945

Zdroj: Vlastní zpracování

Dle ukazatele relativní vzdálenosti od bazální varianty po aplikování metody TOPSIS a konečného pořadí lze konstatovat, že pro modely „Ekonomicky aktivní“ a „Student“ byl doporučen bankovní účet MůjÚčet Plus od Komerční banky. V případě modelu „Senior“ byl stejně jako v předchozích případech určen Osobní účet od Fio banky jako nejlepší varianta.

5.3.4 Metoda váženého součtu

Poslední využitou metodou je metoda váženého součtu, která vychází z principu maximalizace užitku. Následující tabulka s výsledky kromě výsledného pořadí bankovních účtů poskytuje také informaci o celkové hodnotě užitku varianty.

Tabulka 8 - Vyhodnocení bankovních účtů metodou váženého součtu

Banka	Účet	Ekonomicky aktivní		Student		Senior	
		Pořadí	Hodnocení	Pořadí	Hodnocení	Pořadí	Hodnocení
Air Bank	Běžný účet	5	0,53617	10	0,51721	2	0,61166
	Studentský účet			10	0,51721		
Banka CREDITAS	Běžný účet	3	0,55253	5	0,55263	3	0,60049
Česká spořitelna	Studentský účet			11	0,51456		
	Standard účet	12	0,45157	15	0,42946	11	0,46632
	Plus účet	13	0,44510	16	0,42237	12	0,45645
ČSOB	Plus Konto	15	0,33731	18	0,31649	14	0,35831
	Základní účet	17	0,22242	20	0,20124	16	0,24473
Fio banka	Účet pro studenty			**1**	**0,61261**		
	Osobní účet	**1**	**0,58890**	3	0,56606	**1**	**0,66227**
Komerční banka	MůjÚčet Plus	2	0,55919	2	0,56993	4	0,55932
	MůjÚčet Gold	4	0,55078	4	0,56071	5	0,54648
	Studentský účet G2			14	0,46897		
	MůjÚčet	16	0,28842	19	0,27961	15	0,32845
MONETA Money Bank	Tom účet	10	0,49982	8	0,52742	7	0,50829
	Tom účet Plus	10	0,49982	8	0,52742	7	0,50829
	Genius Student			8	0,52742		
	Genius Gold	11	0,48241	13	0,50125	10	0,48666
Raiffeisenbank	Chytrý účet	6	0,52517	6	0,53357	6	0,51064
	Aktivní účet	7	0,52200	7	0,53010	8	0,50580
	Prémiový účet	8	0,52193	9	0,52293	6	0,51064
	Exkluzivní účet	9	0,50899	12	0,50875	9	0,48904
Trinity Bank	Běžný účet	14	0,36822	17	0,34830	13	0,44185
UniCredit Bank	Účet START	18	0,11914	21	0,12122	17	0,14694
	Účet OPEN	19	0,11338	22	0,11491	18	0,13815
	Účet TOP	20	0,09979	23	0,10002	19	0,11741

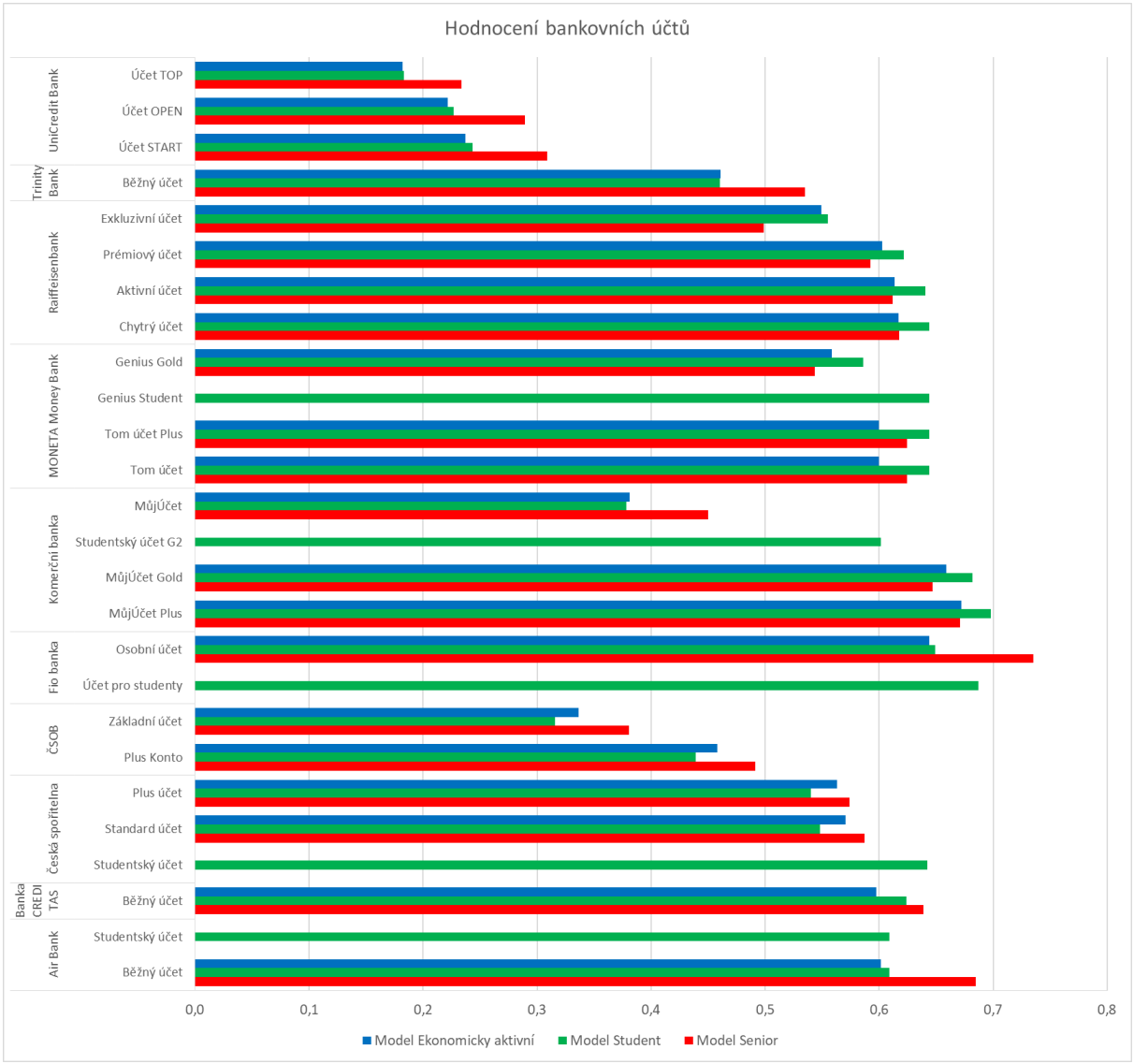
Zdroj: Vlastní zpracování

Poslední ze zvolených metod pro analýzu variant vyhodnotila jako nejvýhodnější bankovní účty ty produkty, které poskytuje Fio banka, a sice běžný účet pro analyzované modely „Ekonomicky aktivní“ a „Senior“ a studentský účet pro model „Student“. Další alternativní varianty jsou v případě modelu „Ekonomicky aktivní“ a „Student“ MůjÚčet Plus od Komerční banky a pro model „Senior“ byl jako druhá nejlepší varianta určen běžný účet od Air bank.

5.4 Diskuze

Jak již bylo zmíněno v kapitole 5.2, tak nejvhodnějšími metodami vícekriteriálního rozhodování pro analýzu bankovních účtů z těch, co byly implementovány, se ukázaly být metoda TOPSIS a metoda váženého součtu, a to zejména z důvodu vhodnosti použití pro rozhodovací problémy, kdy je většina kritérií kvantitativního charakteru. Jmenované metody navíc nejvhodnějším způsobem pracují se vstupními informacemi (tedy váhami kritérií) a také rozhodovateli poskytují podstatnou informaci, která popisuje konečné ohodnocení varianty. Z tohoto důvodu budou shrnuty výsledky těchto dvou metod formou aritmetického průměru hodnocení variant, které byly získány aplikováním těchto dvou metod.

Obrázek 28 - Hodnocení bankovních účtů



Zdroj: Vlastní zpracování

Z výsledků je patrné, že pro analyzované modely „Ekonomicky aktivní“ a „Student“ byl nejlépe vyhodnocen běžný účet MůjÚčet Plus od Komerční banky. V případě modelu Senior byl nejlépe ohodnocen běžný bankovní účet od Fio banky. Je však potřeba zmínit, že jak studentský, tak i běžný bankovní účet od Fio banky byl nadprůměrně ohodnocen narozdíl například od běžného účtu MůjÚčet od Komerční banky, který značně zaostával. V případě doporučených bankovních účtů pro seniory je také potřeba zmínit běžný bankovní účet od Air Bank, který skončil s nadprůměrným ohodnocením, a to z důvodu velmi nízkých poplatků za provádění bankovních operací na pobočce, které má tato banka zpoplatněné podobnou mírou jako Fio banka.

Z grafu lze dále vypozorovat, že nejhoršího ohodnocení nabyly bankovní účty, které nabízí banka UniCredit Bank, a to pro každý zkoumaný model. Jako překvapující se může jevit skutečnost, že jako jedny z nejhůře ohodnocených bankovních účtů dopadly ty, které jsou nabízeny jednou z největších bank v České republice – a sice bankou ČSOB.

K tomuto vyhodnocení bankovních účtů je nutné dodat, že bere v potaz pouze zvolená kritéria. V případě provedení podrobnější analýzy s využitím více kritérií, či s braním v potaz dalších výhod, které některé banky nabízejí (například slevy za frekventovanější využívání účtu), nebo rozšíření množiny variant o další produkty, by výsledky měly pravděpodobně jinou podobu. S tímto je taktéž spjata aktuálnost dat, která rovněž může ovlivnit výsledky analýzy, nejsou-li data aktuální. Je potřeba tedy konstatovat, že zjištěné výsledky nebudou s určitou pravděpodobností v řádech několika měsíců po zveřejnění této práce aktuální.

ZÁVĚR

Dílčím cílem této diplomové práce bylo seznámit čtenáře s problematikou vícekriteriálního rozhodování a analýzy variant. Práce popisuje metody pro stanovení vah kritérií a analýzu variant, které lze využít za různých podmínek a které jsou vhodné pro použití v případě, že jsou známy určité informace. Čtenář byl dále obeznámen s problematikou bankovnictví, která byla primárně zaměřena na bankovní produkty – tedy bankovní účty.

Dalším cílem této práce bylo vytvořit webovou aplikaci, která bude uživateli umožňovat zjednodušení procesu rozhodování při výběru bankovních účtů na základě osobních preferencí za pomoci metod vícekriteriálního rozhodování a analýzy variant. Před zahájením samotné implementace aplikace bylo potřeba seznámit se podrobněji s problematikou vícekriteriálního rozhodování, hodnocení variant a obecně s teorií bankovnictví. Po obeznámení se s touto problematikou bylo pro analýzu bankovních účtů zvoleno několik metod vícekriteriálního rozhodování a stanovení vah kritérií, které bude možné aplikovat.

Za komplikaci lze označit proces sběru dat, kdy každý bankovní subjekt poskytuje informace odlišnou formou, a je tak velmi obtížné zefektivnit proces sběru dat například automatizací, kdy by bylo zapotřebí vytvořit aplikaci, která by dokázala extrahovat požadovaná data z různých typů souborů, zejména však souborů typu PDF. Zároveň je doporučeno, aby sběr dat prováděla osoba (případně tým) dobře znalá této problematice, která by prováděla kontrolu aktuálnosti dat v pravidelných intervalech, jelikož některé banky mohou měnit sazby i několikrát do roka. Data týkající se analyzovaných bankovních účtů této diplomové práce jsou aktuální k datu 18. 2. 2023.

Ověření korektnosti implementovaných metod vícekriteriálního rozhodování bylo provedeno za pomoci několika jednotkových testů, které byly použity jak na data vztahující se k bankovním účtům, tak na data použitá pro demonstraci použití metod pro analýzu variant v odborné literatuře.

Jak již bylo zmíněno v popisu implementace aplikace, tak webový klient byl implementován jako Blazor WebAssembly aplikace. Toto rozhodnutí bylo učiněno z důvodu zajímavé možnosti implementace single-page aplikací pro vývojáře, jejichž preferovaný programovací jazyk je C#. Zároveň se jedná o technologii s potenciálem vyšší využitelnosti v nadcházejících letech, zejména pro full-stack vývojáře softwaru. Serverovou část aplikace tvoří REST aplikační rozhraní, které umožňuje provádění CRUD operací nad daty, autentizuje uživatele

pomocí protokolu OAuth 2.0, poskytuje možnost konfigurace aplikace, a samozřejmě taktéž umožňuje provést analýzu bankovních účtů pomocí nastavené metody pro analýzu variant.

Až při samotné analýze získaných výsledků vyhodnocení variant bylo zjištěno, že pouze některé metody vícekriteriální analýzy variant ze zvoleného výčtu jsou vhodné pro analýzu bankovních účtů. Hlavním důvodem je především podoba dat, případně předpoklady pro použití konkrétní metody (zejména informace týkající se kritérií). Z těchto důvodů je pro analýzu bankovních účtů doporučeno použít metodu TOPSIS nebo metodu váženého součtu. Ačkoli byla provedena analýza na třech vybraných modelech jakožto příkladu hodnocení bankovních účtů, pro přesnější výsledky je vhodné využít manuální způsob zadávání preferencí, kterou umožňuje aplikace provést, jež poskytuje možnost podrobnější analýzy bankovních účtů. Z uvedených výsledků pro modely „Ekonomicky aktivní“, „Student“ a „Senior“ lze vyčíst, že nejlepšího hodnocení pro dospělé a studenty dosahovaly bankovní účty od Komerční banky a Fio banky. Je tomu tak zejména díky nízkým poplatkům za provádění bankovních operací přes internet, bezplatnému vedení účtu a nízkým poplatkům za výběry z bankomatů. Pro občany v penzi byly velmi dobře vyhodnoceny běžné účty, které jsou nabízeny Fio bankou a bankou Air bank, a to zejména z důvodu nízkých poplatků za bankovní operace, které jsou provedené na pobočce banky a nízkých poplatků za výběry z bankomatů.

Jednou z motivací vytvoření aplikace pro analýzu bankovních účtů je skutečnost, že neexistuje příliš nástrojů k tomuto účelu, které využívají analýzu pomocí metod vícekriteriálního rozhodování. Tento fakt byl zjištěn po provedení rešerše v průběhu implementace aplikace. Většina v současné době dostupných aplikací, které také provádějí analýzu bankovních produktů, totiž nevyhodnocuje varianty pomocí metodik vícekriteriálního rozhodování, ale fungují na principu odhadu měsíčních nákladů za využívání bankovního účtu. Avšak i v případě, že autor uvede, že aplikace vyhodnocuje bankovní produkty pomocí metod vícekriteriálního rozhodování, tak lze předpokládat, že konkrétní aplikace umožňuje provést analýzu pomocí jediné metody analýzy variant. Jelikož implementovaná aplikace umožňuje využití více metod pro analýzu, tak lze analyzovat i rozdíly ve vyhodnocení za použití odlišných metod. Z těchto důvodů může být tato práce přínosná v oblasti analýzy bankovních produktů, jelikož oproti konkurenčním nástrojům nabízí širší možnost využití.

Do budoucna by bylo vhodné zamyslet se nad dalším vývojem implementované aplikace. Aplikaci by bylo vhodné upravit takovým způsobem, aby umožňovala provádění podrobnější analýzy bankovních účtů. Znamenalo by to rozšíření datového modelu a získání podrobnějších informací o bankovních účtech. Ačkoli cílem práce nebylo provádět analýzu například

spořících účtů, tak aplikace je implementována tak, že případné rozšíření analýzy o tento typ bankovních účtů obnáší pouze získání dodatečných informací o těchto bankovních produktech.

POUŽITÁ LITERATURA

[1] POLOUČEK, Stanislav. *Bankovnictví*. 2. vyd. V Praze: C.H. Beck, 2013. Beckovy ekonomické učebnice. ISBN 978-80-7400-491-9.

[2] ŠENKÝŘOVÁ, Bohuslava. *Bankovnictví I.* Praha: Grada, 1997. ISBN 80-7169-464-9.

[3] DVOŘÁK, Petr. *Komerční bankovnictví pro bankéře a klienty*. Praha: Linde, 1999. Praktické příručky. ISBN 80-7201-164-2.

[4] TALAŠOVÁ, Jana. *Fuzzy metody vícekriteriálního hodnocení a rozhodování.* Olomouc: Univerzita Palackého, 2003. Monografie. ISBN 80-244-0614-4.

[5] FIALA, Petr. *Modely a metody rozhodování.* 3., přeprac. vyd. V Praze: Oeconomica, 2013. ISBN 978-80-245-1981-4.

[6] ŠUBRT, Tomáš. *Ekonomicko-matematické metody.* Plzeň: Vydavatelství a nakladatelství Aleš Čeněk, 2011. ISBN 978-80-7380-345-2.

[7] *XUnit.net* [online]. © 2023 [cit. 2023-01-08]. Dostupné z: https://xunit.net/

[8] What is a REST API? *IBM* [online]. [cit. 2023-01-08]. Dostupné z: https://www.ibm.com/topics/rest-apis

[9] What is a REST API? *Red Hat* [online]. 8. 5. 2022 [cit. 2023-01-09]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api

[10] JSON Web Token Introduction - jwt.io. *JSON Web Tokens - jwt.io* [online]. Dostupné z: https://jwt.io/introduction

[11] What is OAuth 2.0 and what does it do for you? - Auth0. *Auth0: Secure access for everyone. But not just anyone.* [online]. Copyright © [cit. 10.01.2023]. Dostupné z: https://auth0.com/intro-to-iam/what-is-oauth-2

[12] What is Entity Framework? *Entity Framework Tutorial* [online]. Dostupné z: https://www.entityframeworktutorial.net/what-is-entityframework.aspx

[13] ASP.NET Core Blazor. *Microsoft | Learn* [online]. © 2023, 30.12.2022 [cit. 2023-01-14]. Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-7.0

[14] SKALKOVÁ, Olga. Největší banky v Česku. Nové žebříčky podle klientů a peněz. *Peníze.cz* [online]. 25. 3. 2021 [cit. 2023-01-22]. Dostupné z: https://www.penize.cz/osobni-ucty/432939-nejvetsi-banky-v-cesku-zebricek-podle-poctu-klientu-a-spravovanych-penez

[15] Internet banking on the rise. *Eurostat* [online]. 15. 1. 2018 [cit. 2023-01-22]. Dostupné z: https://ec.europa.eu/eurostat/web/products-eurostat-news/-/DDN-20180115-1

[16] Ceník pro Standard účet České spořitelny. *Česká spořitelna* [online]. 23. 7. 2022, 5 [cit. 2023-01-22]. Dostupné z: https://www.csas.cz/banka/content/inet/internet/cs/cenik_standard_ucet.pdf

[17] Ceník pro Plus účet České spořitelny. *Česká spořitelna* [online]. 23. 7. 2022, 5 [cit. 2023-01-22]. Dostupné z: https://www.csas.cz/banka/content/inet/internet/cs/cenik_plus_ucet.pdf

[18] Sazebník ČSOB pro fyzické osoby – občany. *ČSOB* [online]. 1. 11. 2022, 20 [cit. 2023-01-22]. Dostupné z: https://www.csob.cz/portal/documents/10710/423623/sazebnik-fo-cz.pdf

[19] CENÍK FINANČNÍCH OPERACÍ A SLUŽEB pro fyzické a právnické osoby. *Fio banka* [online]. 5. 8. 2022, 9 [cit. 2023-01-22]. Dostupné z: https://www.fio.cz/docs/cz/cenik_bankovni_sluzby.pdf

[20] Účty pro děti a studenty do 18 let. *Fio banka* [online]. @ 2023 [cit. 2023-01-22]. Dostupné z: https://www.fio.cz/bankovni-sluzby/bankovni-ucty/ucty-pro-mlade

[21] Ceník. *Air Bank* [online]. 28. 11. 2022, 1 [cit. 2023-01-22]. Dostupné z: https://www.airbank.cz/file-download/cenik

[22] Sazebník KB. *Komerční banka* [online]. 1. 1. 2023, 43 [cit. 2023-01-22]. Dostupné z: https://www.kb.cz/getmedia/7e110491-8a9c-443b-9690-cc252eabf43d/kb-sazebnik-obcane.pdf.aspx

[23] CENÍK PRODUKTŮ A SLUŽEB PRO SOUKROMÉ OSOBY. *Raiffeisenbank* [online]. 14. 11. 2022, 7 [cit. 2023-01-22]. Dostupné z: https://www.rb.cz/attachments/ceniky/cenik-pi-1.pdf

[24] Nabídka osobních účtů. *Raiffeisenbank* [online]. 1. 9. 2022, 2 [cit. 2023-01-22]. Dostupné z: https://www.rb.cz/attachments/ceniky/cenik-porovnani-osobnich-uctu.pdf

[25] Sazebník – účty. *Equa bank* [online]. 1. 9. 2022 [cit. 2023-01-22]. Dostupné z: https://www.equabank.cz/download/dms/2236-L003_sazebnik-cz-01092022.pdf

[26] Sazebník odměn za poskytování bankovních služeb: Část fyzické osoby nepodnikající UniCredit Bank Czech Republic and Slovakia, a.s. *UniCredit Bank* [online]. 1. 7. 2022, 27 [cit. 2023-01-22]. Dostupné z: https://www.unicreditbank.cz/content/dam/cee2020-pws-cz/cz-dokumenty/dokumenty-produkty/sazebniky/Sazebnik_pro_fyzicke_osoby.pdf

[27] SAZEBNÍK POPLATKŮ. *Banka CREDITAS* [online]. 14. 9. 2022, 8 [cit. 2023-01-22]. Dostupné z: https://cdn.creditas.cz/image/upload/v1663074160/dokumenty-dulezite-dokumenty/sazebniky-a-dalsi-sdeleni/Sazebn%C3%ADk_poplatk%C5%AF_FON_u%C4%8Dinn%C3%BD_od_14.09.2022_%C4%8Distopis.pdf

[28] Sazebník poplatků za produkty a služby pro fyzické osoby nepodnikatele. *MONETA Money Bank* [online]. 1. 9. 2022, 28 [cit. 2023-01-22]. Dostupné z: https://www.moneta.cz/documents/20143/11740785/mmb-sazebnik-platebni-a-neplatebni-sluzby-fon-01092022.pdf

[29] SOUKALOVÁ, Aneta, Ivan SOUKAL a Jan DRAESSLER. Kalkulátor. *Bankovní poplatky* [online]. @ 2023 [cit. 2023-01-22]. Dostupné z: https://www.bankovnipoplatky.cz/kalkulator.html

[30] Československá obchodní banka, a. s. *Výroční zpráva 2021* [online]. 27. 4. 2022, 339 [cit. 2023-02-21]. Dostupné z: https://www.csob.cz/portal/documents/10710/444804/vz-csob-2021.pdf

[31] Česká spořitelna, a.s. *Výroční zpráva 2021* [online]. 29. 3. 2022, 335 [cit. 2023-02-21]. Dostupné z: https://www.csas.cz/static_internet/cs/Redakce/Ostatni/Ostatni_IE/Prilohy/vz-2021.pdf

[32] Komerční banka, a.s. *Výroční zpráva 2021* [online]. 16. 3. 2022, 337 [cit. 2023-02-21]. Dostupné z: https://www.kb.cz/getmedia/9aafd6ac-7be2-4808-9060-2feae98f9cd0/Vyrocni-zprava-KB-2021_1.pdf.aspx

[33] UniCredit Bank. *Výroční zpráva 2021* [online]. 15. 3. 2022, 256 [cit. 2023-02-21]. Dostupné z: https://www.unicreditbank.cz/content/dam/cee2020-pws-cz/cz-dokumenty/o-bance/vyrocni-zpravy/VZ_2021_CZ_final.pdf

[34] Raiffeisenbank a.s. *Konsolidovaná výroční zpráva 2021* [online]. 21. 4. 2022, 380 [cit. 2023-02-21]. Dostupné z: https://www.rb.cz/attachments/vyrocni-zpravy/Raiffeisenbank_a_s_Konsolidovana_vyrocni_zprava_2021_CZ.pdf

[35] MONETA Money Bank, a.s. *Výroční zpráva 2021* [online]. 21. 3. 2022, 360 [cit. 2023-02-21]. Dostupné z: https://investors.moneta.cz/documents/12270853/20117788/mmb-vyrocni-zprava-2021-cz.pdf

[36] Fio banka, a.s. *Výroční zpráva 2021* [online]. 31. 3. 2022, 85 [cit. 2023-02-21]. Dostupné z: https://www.fio.cz/docs/cz/Fio_banka_vyrocni_zprava_2021.pdf

[37] Air Bank a.s. *Konsolidovaná výroční zpráva* [online]. 17. 3. 2022, 218 [cit. 2023-02-21]. Dostupné z: https://www.airbank.cz/file-download/vyrocni-zprava-2021.pdf

[38] Trinity Bank. *VÝROČNÍ ZPRÁVA* [online]. 28. 3. 2022, 113 [cit. 2023-02-21]. Dostupné z: https://www.trinitybank.cz/file/1354

[39] Banka CREDITAS a.s. *Výroční zpráva 2021* [online]. 12. 4. 2022, 130 [cit. 2023-02-21]. Dostupné z: https://cdn.creditas.cz/image/upload/v1651213454/dokumenty-povinne-uverejnovane-informace/Vyrocni-zprava-2021.pdf

[40] Radzen Blazor Components. *Rapid Application Development for the Web | Radzen* [online]. Copyright © 2016 [cit. 05.02.2023]. Dostupné z: https://www.radzen.com/blazor-components/

[41] DYKSTRA, Tom. Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application (9 of 10). In: *Microsoft | Learn* [online]. 7. 1. 2022 [cit. 2023-03-11]. Dostupné z: https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application

[42] WATMORE, Jason. Blazor WebAssembly – JWT Authentication Example & Tutorial. *Jason Watmore's Blog* [online]. 13. 8. 2020 [cit. 2023-03-12]. Dostupné z: https://jasonwatmore.com/post/2020/08/13/blazor-webassembly-jwt-authentication-example-tutorial

[43] Ceník pro Osobní bankovnictví: fyzické osoby – nepodnikatelé. *Trinity Bank* [online]. 1. 12. 2021, 11 [cit. 2023-03-12]. Dostupné z: https://www.trinitybank.cz/download/1279

[44] BERNASCONI, Claudio. How to Use Serilog in ASP.NET Core Web API. *Claudio Bernasconi – Not only writes code, but solves problems.* [online]. 28. 1. 2022 [cit. 2023-04-02]. Dostupné z: https://www.claudiobernasconi.ch/2022/01/28/how-to-use-serilog-in-asp-net-core-web-api/

[45] JAČKOVÁ, Nikola. *Metody vícekriteriálního rozhodování a jejich aplikace* [online]. Pardubice, 2021 [cit. 2023-04-02]. Dostupné z: https://dk.upce.cz/bitstream/handle/10195/77807/JackovaN_MultikriterialniRozhodovani_AP_2021.pdf. Bakalářská práce. Univerzita Pardubice. Vedoucí práce Mgr. Alena Pozdílková, Ph.D.

PŘÍLOHY

PŘÍLOHA A – TECHNICKÁ DOKUMENTACE

Manuál popisující konfiguraci a nastavení aplikace včetně úkonů nutných k provedení pro spuštění aplikace. V dokumentaci lze také nalézt návod k publikaci a hostování aplikace na Internetové informační službě (IIS) na operačním systému Windows.

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Bankovní kalkulátor – technická dokumentace

Bc. Michal Černota

2023

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

IIS	Internet Information Services
JSON	JavaScript Object Notation
HTML	Hypertext Markup Language

ÚVOD

Tento dokument slouží jako technická dokumentace webové aplikace Bankovní kalkulátor, která byla vytvořena v rámci zpracování diplomové práce na stejné téma v akademickém roce 2022/2023. Dokumentace popisuje postup pro spuštění aplikace, její publikaci a hostování na IIS a konfiguraci aplikace. Text tohoto dokumentu je určen zejména pro uživatele, kteří chtějí aplikaci zprovoznit a používat na svém zařízení.

1 INSTALACE .NET RUNTIME

Před instalací samotné aplikace je potřeba nainstalovat běhové prostředí .NET frameworku. Aplikace vyžaduje instalaci .NET runtime pro .NET 6 a také běhové prostředí pro .NET Core 6. V případě, že tato běhová prostředí nebudou nainstalovaná, tak se aplikaci nepodaří spustit. Identifikovat důvod, proč se aplikaci nepodařilo spustit, lze případně v aplikaci Prohlížeč událostí, což je aplikace operačního systému Windows pro prohlížení například informací, varování a chyb běhu aplikací či jiných komponent operačního systému Windows. Instalační soubory běhového prostředí pro .NET 6 a .NET Core 6 lze nalézt v přiloženém ZIP archivu či na oficiálních stránkách společnosti Microsoft.

Po spuštění instalačního souboru běhového prostředí frameworku .NET 6 *dotnet-runtime-6.0.13-win-x64.exe* se zobrazí následující obrazovka:

Obrázek 29 - Instalace .NET 6 runtime

Zdroj: Vlastní zpracování

Kliknutím na tlačítko Nainstalovat se spustí instalace runtime, která by měla trvat řádově několik sekund. Identický proces je potřeba zopakovat pro instalaci runtime .NET Core 6 spuštěním instalačního souboru *aspnetcore-runtime-6.0.13-win-x64.exe*.

1.1 Kestrel web server

Bude-li aplikace spuštěna pomocí spustitelného souboru BankCalculator.API.exe, pak bude aplikace hostovaná na localhostu prostřednictvím webového serveru Kestrel. Webový server Kestrel je podporován na stejných platformách, jako v případě .NET Core, tedy na všech operačních systémech, které jsou v současnosti nejvíce využívané, konkrétně na operačních systémech Windows, Linux, macOS a iOS.

Kestrel je výchozí webový server pro ASP.NET Core projekty, což znamená, že pokud je na zařízení nainstalováno prostředí .NET 6, tak Kestrel není potřeba dodatečně instalovat.

2 KONFIGURACE APLIKACE

Před spuštěním (respektive instalací aplikace) je potřeba zmínit základní nastavení aplikace. Aplikace lze konfigurovat modifikací konfiguračního souboru *appsettings.json*. Tento JSON soubor vypadá následovně:

Obrázek 30 - Konfigurační soubor aplikace

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "Serilog": {
    "Using": [ "Serilog.Sinks.File", "Serilog.Sinks.Console" ],
    "MinimumLevel": {
      "Default": "Information"
    },
    "WriteTo": [
      {
        "Name": "File",
        "Args": {
          "path": "logs/webapi~.log",
          "rollingInterval": "Day",
          "outputTemplate": "[{Timestamp:yyyy-MM-dd HH:mm:ss.fff zzz} {CorrelationId} {Level:u3}] {Username} {Message:lj}{NewLine}{Exception}",
          "encoding": "System.Text.Encoding::UTF8"
        }
      },
      {
        "Name": "Console"
      }
    ]
  },
  "Security": {
    "Secret": "h0SHehONcezbFqDTeKrtHLbc0p0NAyzAjdczSNiorWmLjqMXQF"
  },
  "DecisionAnalysis": {
    "DecisionMethodType": "WeightedSumMethod"
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "Dev": "server=localhost;port=3306;database=bank;uid=root;password=root"
  },
  "UseInMemoryDatabase": true
}
```

Zdroj: Vlastní zpracování

Na obrázku výše lze vidět nastavení pro logování aplikace, nastavení zabezpečení, aktuálně nastavenou metodu vícekriteriálního rozhodování pro analýzu bankovních účtů, filtr pro požadavky, a nastavení datového uložiště. Z uvedeného nastavení je doporučeno konfigurovat pouze nastavení pro připojení k datovému uložišti, případně logování aplikace. Je-li nastavena hodnota *UseInMemoryDatabase* na hodnotu *true*, pak aplikace ukládá data do databáze, která je vytvořena v rámci paměťového prostoru aplikace, přičemž případná modifikace dat po spuštění aplikace je po ukončení aplikace ztracena, a po opětovném spuštění jsou data ve výchozím stavu. Z toho vyplývá fakt, že tento režim slouží pouze k testovacím účelům, pokud uživatel nemůže využít databázi MySQL. V případě používání aplikace v produkčním prostředí je doporučeno zajistit databázové uložiště MySQL a v nastavení aplikace nakonfigurovat connection string ve stejnojmenném atributu JSON souboru. Zároveň je potřeba nastavit hodnotu *UseInMemoryDatabase* na hodnotu *false*.

2.1 Popis datových souborů

Při prvotním spuštění aplikace dojde kromě vytvoření databázového schématu a databázových tabulek také k vytvoření patřičných datových entit, tedy vložení záznamů do vytvořených databázových tabulek. Tato data se získávají z datových souborů aplikace *bankAccounts.json*, *banks.json* a *users.json*. Z názvů těchto souborů lze vyvodit, že obsahují data, která se vztahují k bankovním účtům, bankám a uživatelům. Je doporučeno v pravidelných intervalech aktualizovat datové soubory, které uchovávají informace o bankovních účtech a bankách. Je-li vyžadováno při prvotním spuštění aplikace vytvořit výchozí účet administrátora, pak je před spuštěním potřeba tento účet definovat v odpovídajícím souboru *users.json*.

Obrázek 31 - Ukázka datového souboru


```
[
  {
    "Id": 1,
    "BankId": 1,
    "Name": "Standard účet",
    "BankAccountType": "Current",
    "InterestRate": 0,
    "HasInternetbanking": true,
    "AccountManagementFees": {
      "Id": 1,
      "AccountCreationFee": 0,
      "AccountManagementFee": 0
    },
    "AccountWithdrawalFees": {
      "Id": 1,
      "AtmWithdrawalFee": 0,
      "WithdrawalFromOtherBanksAtmFee": 40,
      "AtmWithdrawalAbroadFee": 5,
      "WithdrawalFromOtherBanksAtmAbroadFee": 40,
      "OfficeWithdrawalFee": 100
    },
    "AccountInfoFees": {
      "Id": 1,
      "AtmBalanceInquiryFee": 0,
      "OtherBanksAtmBalanceInquiryFee": 20,
      "InformativeSmsFee": 2.5,
      "ElectronicalAccountStatementFee": 0,
      "PostAccountStatementFee": 50
    },
    "AccountPaymentFees": {
      "Id": 1,
      "OutgoingPaymentInternetFee": 0,
      "OutgoingPaymentOfficeFee": 100,
      "EstablishmentOfStandingOrderInternetFee": 0,
      "ExecutionOfStandingOrderInternetFee": 0,
      "EstablishmentOfStandingOrderOfficeFee": 100,
      "ExecutionOfStandingOrderOfficeFee": 0
    }
  },
```


Zdroj: Vlastní zpracování

2.2 Logovací soubor

Aplikace ve výchozím režimu loguje do logovacích souborů, které lze nalézt v podadresáři *logs*, přičemž aplikace vytváří samostatný soubor pro každý den. Tyto logovací soubory poskytují informace o běhu aplikace, jako například informace o provedených dotazech, ukládání entit, vyhodnocení bankovních účtů, směrování aplikace atd., a jsou tak velmi důležité zejména pro analýzu chybových stavů aplikace. Níže lze vidět příklad obsahu logovacího souboru:

Obrázek 32 - Logovací soubor aplikace

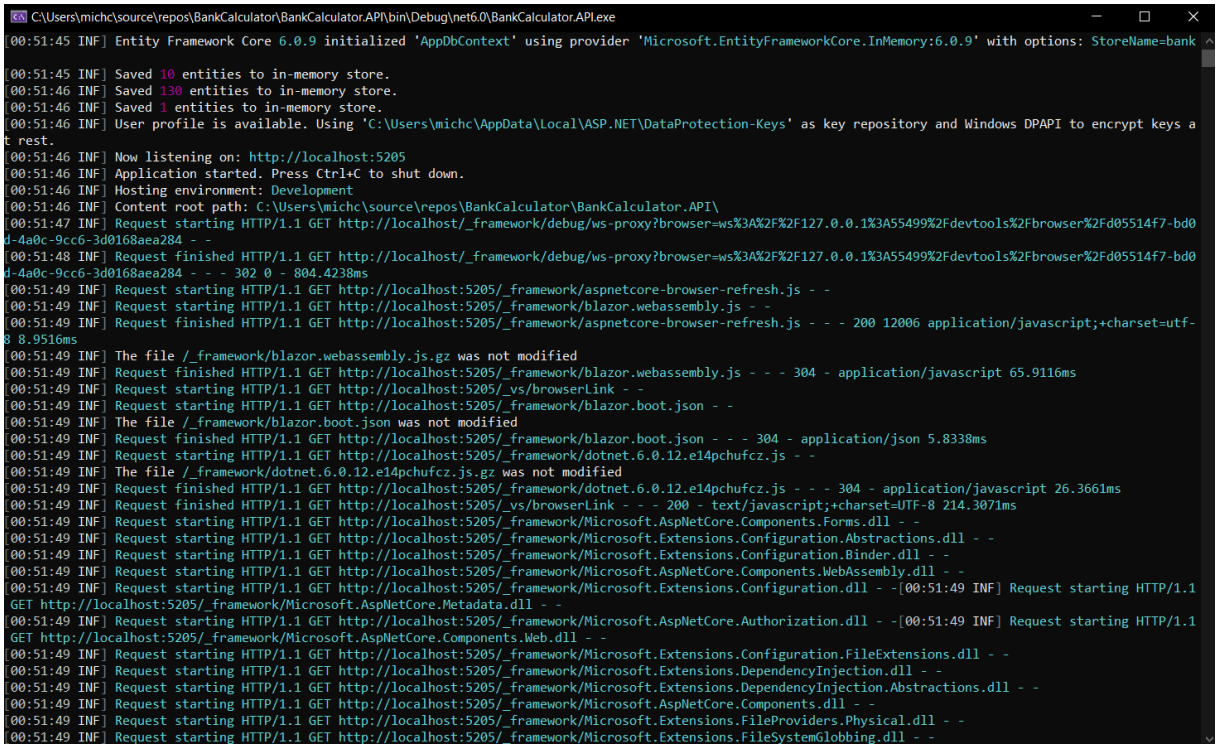
```
[2023-01-28 23:10:39.798 +01:00  INF]  Entity Framework Core 6.0.9 initialized 'AppDbContext' using provider 'Microsoft.EntityFrameworkCore.InMemory:6.0.9' with options: StoreName=bank
[2023-01-28 23:10:40.194 +01:00  INF]  Saved 10 entities to in-memory store.
[2023-01-28 23:10:40.349 +01:00  INF]  Saved 125 entities to in-memory store.
[2023-01-28 23:10:40.556 +01:00  INF]  Saved 1 entities to in-memory store.
[2023-01-28 23:10:40.604 +01:00  INF]  User profile is available. Using 'C:\Users\michc\AppData\Local\ASP.NET\DataProtection-Keys' as key repository and Windows DPAPI to encrypt keys at re
[2023-01-28 23:10:40.759 +01:00  INF]  Now listening on: http://localhost:5000
[2023-01-28 23:10:40.761 +01:00  INF]  Now listening on: https://localhost:5001
[2023-01-28 23:10:40.765 +01:00  INF]  Application started. Press Ctrl+C to shut down.
[2023-01-28 23:10:40.766 +01:00  INF]  Hosting environment: Production
[2023-01-28 23:10:40.766 +01:00  INF]  Content root path: C:\Users\michc\source\repos\BankCalculator\BankCalculator.API\bin\Release\net6.0\publish\
[2023-01-28 23:10:52.170 +01:00  INF]  Request starting HTTP/1.1 GET http://localhost:5000/_framework/blazor.webassembly.js - -
[2023-01-28 23:10:52.226 +01:00  INF]  The file /_framework/blazor.webassembly.js.br was not modified
[2023-01-28 23:10:52.232 +01:00  INF]  Request finished HTTP/1.1 GET http://localhost:5000/_framework/blazor.webassembly.js - - - 304 - application/javascript 68.4388ms
[2023-01-28 23:10:52.248 +01:00  INF]  Request starting HTTP/1.1 GET http://localhost:5000/_framework/blazor.boot.json - -
[2023-01-28 23:10:52.262 +01:00  INF]  Sending file. Request path: '/_framework/blazor.boot.json.br'. Physical path: 'C:\Users\michc\source\repos\BankCalculator\BankCalculator.API\bin\Rele
[2023-01-28 23:10:52.265 +01:00  INF]  Request finished HTTP/1.1 GET http://localhost:5000/_framework/blazor.boot.json - - - 200 2932 application/json 16.4738ms
[2023-01-28 23:10:52.278 +01:00  INF]  Request starting HTTP/1.1 GET http://localhost:5000/_framework/dotnet.6.0.12.e14pchufcz.js - -
[2023-01-28 23:10:52.285 +01:00  INF]  The file /_framework/dotnet.6.0.12.e14pchufcz.js.br was not modified
[2023-01-28 23:10:52.286 +01:00  INF]  Request finished HTTP/1.1 GET http://localhost:5000/_framework/dotnet.6.0.12.e14pchufcz.js - - - 304 - application/javascript 8.1613ms
[2023-01-28 23:11:02.196 +01:00  INF]  Request starting HTTP/1.1 POST http://localhost:5000/api/BankAccount/RankBankAccounts application/json;+charset=utf-8 829
[2023-01-28 23:11:02.225 +01:00  INF]  Executing endpoint 'BankCalculator.API.Controllers.BankAccountController.RankBankAccounts (BankCalculator.API)'
[2023-01-28 23:11:02.253 +01:00  INF]  Route matched with {action = "RankBankAccounts", controller = "BankAccount"}. Executing controller action with signature Microsoft.AspNetCore.Mvc.Act
[2023-01-28 23:11:02.499 +01:00  INF]  Weighted criteria: [{"Name":"Index spolehlivosti banky","Weight":0.011235955056179775,"CriterionType":"Maximizing","$type":"Criterion"},{"Name":"Inte
[2023-01-28 23:11:02.594 +01:00  INF]  Entity Framework Core 6.0.9 initialized 'AppDbContext' using provider 'Microsoft.EntityFrameworkCore.InMemory:6.0.9' with options: StoreName=bank
[2023-01-28 23:11:02.851 +01:00  INF]  Unsitisfied criteria: [{"Name":"Poplatek za zřízení trvalé platby přes Internet","Weight":0.07865168539325842,"CriterionType":"Minimizing","$type":"C
[2023-01-28 23:11:02.896 +01:00  INF]  Decision result: [{"Bank":"Fio banka","BankAccount":"Osobní účet","Rank":1,"Evaluation":0.63562},{"Bank":"Equa bank","BankAccount":"Běžný účet","Rank
[2023-01-28 23:11:02.982 +01:00  INF]  Executing OkObjectResult, writing value of type 'BankCalculator.Shared.DTOs.BankAccount.BankAccountEvaluation.BankAccountsEvaluationResultDto'.
[2023-01-28 23:11:02.998 +01:00  INF]  Executed action BankCalculator.API.Controllers.BankAccountController.RankBankAccounts (BankCalculator.API) in 732.0478ms
[2023-01-28 23:11:03.002 +01:00  INF]  Executed endpoint 'BankCalculator.API.Controllers.BankAccountController.RankBankAccounts (BankCalculator.API)'
[2023-01-28 23:11:03.006 +01:00  INF]  Request finished HTTP/1.1 POST http://localhost:5000/api/BankAccount/RankBankAccounts application/json;+charset=utf-8 829 - 200 - application/json;+c
[2023-01-28 23:11:04.728 +01:00  INF]  Request starting HTTP/1.1 GET http://localhost:5000/api/BankAccount/17 - -
[2023-01-28 23:11:04.738 +01:00  INF]  Executing endpoint 'BankCalculator.API.Controllers.BankAccountController.Get (BankCalculator.API)'
[2023-01-28 23:11:04.753 +01:00  INF]  Route matched with {action = "Get", controller = "BankAccount"}. Executing controller action with signature Microsoft.AspNetCore.Mvc.ActionResult`1[B
[2023-01-28 23:11:04.895 +01:00  INF]  Entity Framework Core 6.0.9 initialized 'AppDbContext' using provider 'Microsoft.EntityFrameworkCore.InMemory:6.0.9' with options: StoreName=bank
[2023-01-28 23:11:05.052 +01:00  INF]  Executing OkObjectResult, writing value of type 'BankCalculator.Shared.DTOs.BankAccount.BankAccountDetailDto'.
[2023-01-28 23:11:05.070 +01:00  INF]  Executed action BankCalculator.API.Controllers.BankAccountController.Get (BankCalculator.API) in 298.0693ms
[2023-01-28 23:11:05.077 +01:00  INF]  Executed endpoint 'BankCalculator.API.Controllers.BankAccountController.Get (BankCalculator.API)'
[2023-01-28 23:11:05.082 +01:00  INF]  Request finished HTTP/1.1 GET http://localhost:5000/api/BankAccount/17 - - - 200 - application/json;+charset=utf-8 354.4452ms
```

Zdroj: Vlastní zpracování

3 SPUŠTĚNÍ APLIKACE

Webovou aplikaci Bankovní kalkulátor lze spustit spuštěním souboru *BankCalculator.API.exe*. Aplikace bude hostována na webovém serveru Kestrel a dojde k uložení entit, které jsou definovány v datových souborech, do nakonfigurovaného uložiště dat. Po spuštění aplikace by se zobrazí konzole s následujícím výstupem:

Obrázek 33 - Zobrazené okno konzole po spuštění aplikace



Zdroj: Vlastní zpracování

Z výstupu lze mimo jiné vypozorovat, na jaké adrese je aplikace spuštěna, tedy na adrese http://localhost:5205. Spustíte-li webový prohlížeč (podporovány jsou současné verze prohlížečů Google Chrome, Mozilla Firefox, Safari a Microsoft Edge), do jehož adresního řádku zadáte tuto adresu, pak se zobrazí hlavní stránka aplikace.

4 PUBLIKACE APLIKACE NA IIS

Aplikaci je samozřejmě možné publikovat na různé platformy, jako například Heroku či AWS. Bude předvedena ukázka nasazení aplikace na IIS (Internet Information Service) na operačním systému Windows 10.

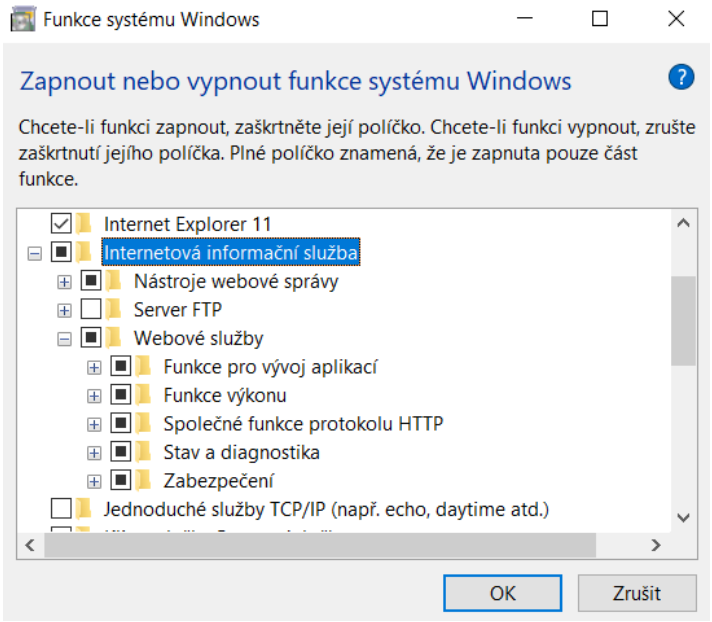
Nejdříve je doporučeno provést již zmíněnou konfiguraci aplikace (zejména konfiguraci datového uložiště), kterou lze provést v souboru *appsettings.json* a zároveň je potřeba nainstalovat běhové prostředí pro .NET 6 a .NET Core 6, jejichž instalace již byla taktéž popsána v tomto dokumentu.

Pro hostování aplikace prostřednictvím IIS je navíc potřeba nainstalovat ASP.NET Core IIS modul, který lze rovněž nalézt na oficiálních stránkách společnosti Microsoft.

4.1 Povolení služby IIS

Dále je potřeba povolit službu IIS v operačním systému Windows, která je ve výchozím nastavení vypnuta. Přejděte do nastavení „Ovládací panely“, dále zvolte nabídku „Programy“. Pokračujte zvolením možnosti „Zapnout nebo vypnout funkce systému Windows“. V nabídce funkcí vyhledejte „Internetová informační služba“. Zvolte tuto funkci zaškrtnutím checkboxu. Mělo by se nastavit výchozí nastavení IIS. Po zobrazení podřízeného nastavení funkce „Internetová informační služba“ bude nabídka vypadat následovně:

Obrázek 34 - Povolení funkce IIS



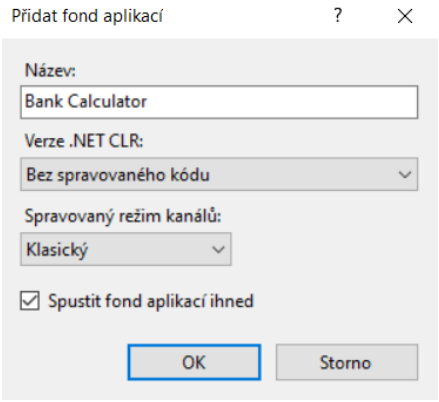
Zdroj: Vlastní zpracování

Klikněte na tlačítko „OK“ a operační systém provede požadované změny. Po dokončení konfigurace vyhledejte aplikaci „Správce internetové informační služby v nabídce „Start“ a aplikaci spusťte.

4.2 Vytvoření aplikačního fondu a aplikace

V aplikace klikněte pravým tlačítkem na možnost „Fondy aplikací“ a zvolte „Přidat fond aplikací“. V dialogovém okně zadejte název aplikačního poolu, pro verzi .NET CLR zvolte možnost „Bez spravovaného kódu“ a jako spravovaný režim kanálů zvolte režim „Klasický“. Ponechte volbu pro okamžité spuštění poolu a potvrďte nastavení.

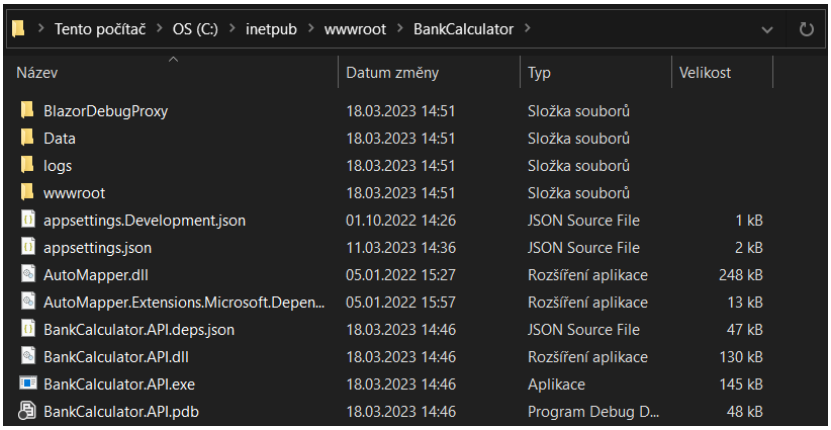
Obrázek 35 - Vytvoření fondu aplikací



Zdroj: Vlastní zpracování

Nyní vytvořte nový podadresář v adresáři *C:\inetpub\wwwroot*. V rámci ukázky bude vytvořen podadresář s názvem *BankCalculator*. Do vytvořeného adresáře nakopírujte soubory publikovaní aplikace Bankovní kalkulátor.

Obrázek 36 - Adresář aplikace



Zdroj: Vlastní zpracování

Aktualizujete-li v aplikaci „Správce Internetové informační služby“ webovou stránku „Default Web Site“, zobrazí se se stromové struktuře vytvořený adresář aplikace. Na vytvořený adresář klikněte pravým tlačítkem myši a zvolte „Převést na aplikaci“. V dialogovém okně vytvořte vytvořený aplikační pool a potvrďte.

Nyní je potřeba modifikovat defaultní soubor webové aplikace *index.html*, který se nachází v podadresáři *wwwroot* webové aplikace. Zde je potřeba v hlavičce HTML stránky doplnit název vytvořeného adresáře do atributu *href* elementu *base*, jinak bude aplikace po pokusu o spuštění vracet chybový status kód 404, jelikož se nepodaří nalézt soubory potřebné pro běh aplikace. Cesta odkazu musí být ukončená lomítkem.

Obrázek 37 - Modifikace souboru index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no" />
    <title>Bankovní kalkulátor</title>
    <base href="/BankCalculator/" />
    <link href="css/bootstrap/bootstrap.min.css" rel="stylesheet" />
    <link href="css/app.css" rel="stylesheet" />
    <link href="BankCalculator.Client.styles.css" rel="stylesheet" />
    <link rel="stylesheet" href="_content/Radzen.Blazor/css/material-base.css">
</head>

<body>
    <div id="app">
        <div class="vh-100 d-flex align-items-center justify-content-center">
            <div class="h5 spinner-border text-primary" style="width: 4rem; height: 4rem;"></div>
        </div>
    </div>

    <div id="blazor-error-ui">
        An unhandled error has occurred.
        <a href="" class="reload">Reload</a>
        <a class="dismiss">✕</a>
    </div>
    <script src="_framework/blazor.webassembly.js"></script>
    <script src="_content/Radzen.Blazor/Radzen.Blazor.js"></script>
</body>

</html>
```

Zdroj: Vlastní zpracování

Po spuštění webového prohlížeče a následném zadání adresy *http://localhost/BankCalculator* do adresního řádku prohlížeče se zobrazí hlavní stránka aplikace.

ZÁVĚR

Čtenář této dokumentace byl obeznámen s požadavky pro spuštění aplikace Bankovní kalkulátor a její konfigurací. Dále byla demonstrována ukázka spuštění aplikace a její hostování na webovém serveru Kestrel. Rovněž byla čtenáři demonstrován proces vystavení aplikace na službu Internetové informační služby, vytvoření aplikačního poolu a vytvoření webové aplikace v administrativní aplikaci Internetové informační služby.