UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKA

BAKALÁŘSKÁ PRÁCE

2025 Patrick Karal

Univerzita Pardubice

Fakulta Elektrotechniky a Informatiky

Webová aplikace pro správu financí

Bakalářská práce

2025 Patrick Karal

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Patrick Karal**
Osobní číslo:	**I22248**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Webová aplikace pro správu financí**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem práce je vytvořit webovou aplikaci pro správu osobních financí. Aplikace umožní záznam příjmů i výdajů. V aplikaci bude umožněno zobrazovat finanční přehledy ve formě grafů. Jednotlivé příjmy i výdaje budou kategorizovány.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

JEMEROV, Dmitry a ISAKOVA, Svetlana. *Kotlin in action*. Shelter Island: Manning, [2017]. ISBN 978-1--61729-329-0
LACKO, Ľuboslav. *Mistrovství – Android*. Computer Press, 2017. ISBN 978-80-251-4875-4.

Vedoucí bakalářské práce: **Ing. Jan Panuš, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2024**
Termín odevzdání bakalářské práce: **16. května 2025**

L.S.

prof. Ing. Petr Doležel, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2025

Prohlašuji:

Práci s názvem Webová aplikace pro správu financí 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., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, 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. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 04. 05. 2025

Patrick Karal

PODĚKOVÁNÍ

Chtěl bych poděkovat své rodině za podporu při psaní této práce a panu profesoru Panušovi.

ANOTACE

Cílem bakalářské práce je vytvořit webovou aplikaci, která bude zajišťovat možnost správy osobních financí a jejich zpracování. Hlavní funkce je možnost zobrazení grafů, které umožní vizualizaci financí, vedlejší funkce je automatické vytváření geografických dat a jejich zobrazení podle toho, kde byla transakce uskutečněna. Teoretická část se bude věnovat vývoji webových aplikací a praktická část implementací konkrétní webové aplikace.

KLÍČOVÁ SLOVA

Webová aplikace, C#,SQLlite,MVC

ANNOTATION

The aim of the bachelor's thesis is to create a web application that will provide the possibility of managing personal finances and their processing. The main function is the ability to display graphs that will allow visualization of finances, a secondary function is the automatic creation of geographic data and their display depending on where the transaction was made. The theoretical part will be devoted to the development of web applications. The practical part will deal with the implementation of a specific web application.

KEYWORDS

Web application, C#, SQL Lite, MVC

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

API – Application Programming Interface, rozhraní pro komunikaci mezi aplikacemi

CSV – Comma-Separated Values, formát textového souboru s oddělenými hodnotami

C# – Programovací jazyk vyvinutý firmou Microsoft

EF – Entity Framework, objektově-relační mapovač pro práci s databází v .NET

HTML – HyperText Markup Language, značkovací jazyk pro tvorbu webových stránek

HTTP – HyperText Transfer Protocol, protokol pro přenos dat mezi klientem a serverem

JSON – JavaScript Object Notation, formát pro výměnu dat

MVC – Model-View-Controller, architektonický vzor pro strukturování aplikací

SQL – Structured Query Language, jazyk pro práci s relačními databázemi

URL – Uniform Resource Locator, adresa zdroje na internetu

WWW – World Wide Web, soustava propojených dokumentů na internetu

ÚVOD

Cílem bakalářské práce je vytvořit webovou aplikaci pro správu osobních financí. Aplikace bude umožnovat nahrát data ve formátu CSV a poté v nich filtrovat, odebírat, přidávat a upravovat. Zároveň bude obsahovat funkci, která umožní automatické vytvoření polohy každé provedené transakce. Všechny transakce bude možné zobrazit v grafech a v těchto grafech poté i filtrovat.

K vypracování bakalářské práce jsem se rozhodl k využití vývojového prostředí Visual Studio. Využité technologie byly ASP.NET Core MVC společně s SQL Lite a API od Nominatee a Chart.js.

1 TEORETICKÁ ČÁST

1.1 Vývoj webové aplikace

Než se začne vyvíjet webová aplikace, je potřeba si zvolit, zda je zadání vhodné k webovému prostředí nebo spíše k nativní aplikaci. Webové aplikace mají hlavní nevýhodu v tom, že nemohou pracovat bez připojení k internetu. Pokud tedy víme, že aplikace bude potřebovat připojení k internetu, záleží pak už jen na tom, jestli bude výkonově náročná. V případě, že jsme se rozhodli pro webovou aplikaci, musíme zvolit framework, ve kterém bude aplikace vytvořena. Na výběr máme z několika možností. Jedná se o ASP.NET Core, Vue.js, React, Django a mnoho dalších. Volba frameworku záleží na tom, zda už využíváme jiné technologie a jaké bude využití aplikace. [1]

1.1.1 World Wide Web

World Wide web neboli známý pod zkratkou WWW je souhrn webových stránek na světové informační síti, ke kterým se připojujeme pomocí internetu. Byl založen roku 1983 panem Tim Berners-Lee a spravován organizací CERN (od roku 1989 až do 1994), dnes je spravován organizací W3C založenou v roce 1994. Důvod založení byla potřeba šíření výsledku studií univerzit na celém světe. V dnešní době se využívá pro sdílení videí, nakupování na internetových obchodech, online komunikaci, práci s daty nebo hraní her.[2]

1.1.2 Princip fungování webových aplikací

Webové aplikace jsou softwarové aplikace, které běží na webových serverech k nimž se uživatelé připojují pomocí webového prohlížeče. V porovnání s nativní aplikací není potřeba je instalovat na používané zařízení. Stačí mít internetové připojení a webový prohlížeč kompatibilní s aplikací. Tyto aplikace obvykle využívají architekturu klient-server, která se skládá ze dvou částí. [3]

Frontend, tedy klientská část aplikace, má na starosti zobrazování uživatelského rozhraní a umožňuje uživateli s aplikací pracovat. K jeho vývoji se používají technologie jako HTML, CSS a JavaScript, které určují vzhled a chování stránek. Moderní webové aplikace často využívají pokročilé frameworky, například Angular nebo React, jež usnadňují tvorbu dynamických a responzivních rozhraní. Backend, tedy serverová část, zajišťuje zpracování uživatelských požadavků, komunikaci s databází a řízení logiky celého systému. K jeho tvorbě se nejčastěji využívají jazyky jako C# nebo Java, které umožňují vytvářet spolehlivé a bezpečné serverové aplikace. [4]

1.1.3 Rozdíl mezi statickými a dynamickými webovými stránkami

Statické webové stránky jsou stránky neměnné, tedy jejich obsah je pořád stejný a není závislý na žádných uživatelských vstupech. Každá stránka je samostatný soubor na serveru, který si klient při načtení automaticky stáhne. [5]

Výhody statických webových stránek:

Statické webové stránky vynikají především rychlostí načítání, protože se obsah nemusí dynamicky generovat – prohlížeč si jednoduše stáhne hotové HTML soubory. Díky tomu, že nevyžadují připojení k databázi ani validaci dat, jsou technicky nenáročné a snadno se spravují. Jejich tvorba je jednoduchá – vystačí si s HTML a CSS, případně s trochou JavaScriptu pro základní interaktivitu. Výhodou je také vyšší úroveň bezpečnosti, protože absence složité serverové logiky a databází snižuje riziko útoků, jako je například SQL Injection.

Nevýhody statických webových stránek:

Údržba statických stránek může být časově náročná, protože každá stránka existuje jako samostatný soubor – jakákoli změna (např. v hlavičce) se musí provádět ručně na všech stránkách zvlášť.

Tyto stránky také neposkytují žádnou pokročilou interakci s uživatelem, protože nereagují na vstupy nebo personalizaci obsahu.

Kvůli těmto omezením nejsou statické weby vhodné pro rozsáhlé projekty s desítkami či stovkami stránek – jejich správa se pak stává neefektivní a náchylná k chybám.

Příklady statických webových stránek:

Mezi časté příklady patří osobní portfolia, která slouží k prezentaci dovedností a projektů.

Dále také prezentační weby, které poskytují základní informace o produktu, službě nebo události.

Informační weby, například s návody nebo dokumentací, využívají jednoduchou strukturu a rychlé načítání.

V neposlední řadě zde můžeme zahrnout i jednoduché firemní weby, kde stačí zobrazit informace jako kontakty, otevírací dobu nebo nabídku služeb.

Dynamické webové stránky na rozdíl od statických generují obsah v reálném čase na uživatelských požadavcích nebo uložených dat v databázi. Jsou řízeny backendem a uživateli umožnují interakci se stránkou, například přihlašování, generování reportů, zobrazování obsahu, odesílání formulářů apod.

Výhody dynamických webových stránek:

Dynamické weby umožňují interaktivitu, takže uživatelé mohou zadávat data, odesílat formuláře nebo měnit zobrazený obsah v reálném čase. Díky tomu, že se obsah generuje automaticky, je správa obsahu mnohem efektivnější, což ocení hlavně weby s velkým množstvím informací. Výhodou je také personalizace – uživatelé mohou vidět obsah přizpůsobený svým preferencím, například doporučené produkty v e-shopu. Dynamické stránky jsou navíc napojeny na databázi (např. SQL Server, MySQL, PostgreSQL), což umožňuje ukládat, aktualizovat a vyhledávat data podle potřeby.

Nevýhody dynamických webových stránek:

Dynamické weby mají vyšší nároky na výkon, protože při každém požadavku musí server stránku zpracovat a vygenerovat, což může zpomalit její načítání. Po bezpečnostní stránce jsou více náchylné k útokům, například SQL Injection nebo Cross-Site Scripting, pokud nejsou správně zabezpečené. Jejich vývoj je také složitější, protože vyžaduje znalost pokročilejších programovacích jazyků a frameworků, jako jsou C# (ASP.NET Core MVC), PHP, Python (Django) nebo JavaScript (např. Node.js).

Příklad dynamických webový:

Typickým příkladem jsou internetové obchody jako Alza nebo Allegro, kde se obsah mění podle uživatele – například nabídky, košík nebo stav objednávky. Sociální sítě typu Facebook nebo Twitter umožňují sdílení příspěvků, komentáře a notifikace, které se přizpůsobují každému uživateli. Dále sem patří online hry, například IO hry, které reagují na akce hráče v reálném čase. Mezi dynamické weby patří i streamovací platformy jako Netflix, Voyo nebo HBO, kde se zobrazují personalizovaná doporučení a přehrávání je řízeno podle uživatelského účtu. Ve výsledku je tedy potřeba si uvědomit jaký typ stránky budeme vytvářet. V případě, že chceme stránky, které dovolí uživatelům interakci s daty, které se na stránce nacházejí, je potřeba zvolit dynamický typ (zobrazování reportů). Pokud ale chceme udělat třeba stránku pro restauraci ohledně jejich menu, tak bohatě stačí statický typ.

1.2 Webové technologie

1.2.1 Možné technologie pro vývoj

Při vývoji webové aplikace je důležité si zvolit vhodnou technologie, která zajistí efektivní výkon, škálovatelnost a bezpečnost aplikace. Volba technologií závisí na mnoha faktorech, jakými jsou požadavky na funkcionalitu, dostupnost, cena vývoje, předem používané technologie náročnosti na výkon a počtu uživatelů.

1.2.2 Backendové technologie

Mezi oblíbené technologie pro serverovou část webových aplikací patří ASP.NET Core MVC, moderní framework od Microsoftu pro jazyk C#. Podporuje architekturu MVC (Model-View-Controller) a je vhodný pro tvorbu výkonných a bezpečných webových aplikací.

Další možností je Node.js spolu s Express.js, což je backendové řešení postavené na JavaScriptu. Výhodou je, že celý vývoj (frontend i backend) může probíhat v jednom jazyce – JavaScriptu.

Django je výkonný a bezpečný framework v jazyce Python, který zrychluje vývoj díky svému přístupu „batteries included“ – nabízí spoustu funkcí hned po instalaci.

Pro vývoj rozsáhlých a robustních aplikací v jazyce Java se využívá Spring Boot, který zjednodušuje konfiguraci a umožňuje rychlejší nasazení aplikací.

1.2.3 Frontendové technologie

Mezi technologie pro tvorbu uživatelského rozhraní patří například Razor Pages, což je šablonovací systém integrovaný do ASP.NET Core MVC. Umožňuje propojení HTML s backendovou logikou v jazyce C# a je vhodný pro jednodušší i středně složité aplikace.[6]

React.js je populární JavaScriptový framework založený na komponentách, který se používá k vytváření rychlých a interaktivních uživatelských rozhraní.

Angular je komplexní frontendový framework od Googlu, postavený na TypeScriptu, a je vhodný pro rozsáhlejší aplikace s větší strukturou.

Lehčí alternativou je Vue.js, který nabízí jednoduchou syntaxi, snadné učení a flexibilitu při vývoji menších až středně velkých projektů.

Mezi další používané nástroje pro tvorbu vzhledu a rozvržení aplikace patří také **Bootstrap** – open-source CSS framework, který usnadňuje tvorbu responzivního a vizuálně konzistentního uživatelského rozhraní.

1.2.4 Databázové technologie

Microsoft SQL Server je výkonný relační databázový systém od Microsoftu, který se skvěle integruje s ASP.NET Core. Nabízí široké možnosti, jako jsou transakce, uložené procedury (stored procedures) a pokročilé zabezpečení. Díky těmto vlastnostem je vhodný pro rozsáhlé podnikové aplikace a komplexní webové systémy.[7]

PostgreSQL je open-source databáze známá svou stabilitou, výkonností a podporou pokročilých analytických funkcí. Umožňuje ukládat data jak ve strukturovaném formátu (SQL), tak i v nestrukturovaném formátu (JSONB), což ji předurčuje k využití v datových a analytických aplikacích.

MySQL patří mezi nejrozšířenější relační databázové systémy. Je ceněn pro svou jednoduchost, rychlost a širokou podporu. Často se používá v e-commerce řešeních, CMS systémech (např. WordPress) a v různých webových aplikacích.

SQLite je lehká relační databáze, která nevyžaduje žádný samostatný server – veškerá data jsou uložena v jediném souboru. Díky tomu je vhodná pro mobilní aplikace, vestavěné systémy a menší webové projekty, kde je jednoduchost a nenáročnost prioritou.[8]

1.2.5 Výběr backendové technologie

Pro vývoj aplikace byla po zvážení dostupných možností zvolena technologie ASP.NET Core MVC. Hlavním důvodem byla předchozí zkušenost s touto platformou získaná během výuky, což umožnilo rychlejší a efektivnější začátek vývoje.

Velkou výhodou je použití architektury Model-View-Controller, která přirozeně odděluje uživatelské rozhraní od aplikační logiky. Díky tomu je kód lépe čitelný, přehlednější, snadněji se udržuje a rozšiřuje. Oddělení backendu od frontendu navíc zvyšuje flexibilitu při vývoji.

Dalším přínosem je podpora pro unit testování – oddělená logika aplikace umožňuje jednodušší psaní testů, což přispívá ke stabilitě a spolehlivosti celého systému. ASP.NET Core MVC také poskytuje plnou kontrolu nad HTML a CSS výstupem, což je výhoda oproti starším technologiím, jako jsou WebForms, kde je výstup generován automaticky a méně přizpůsobitelný.

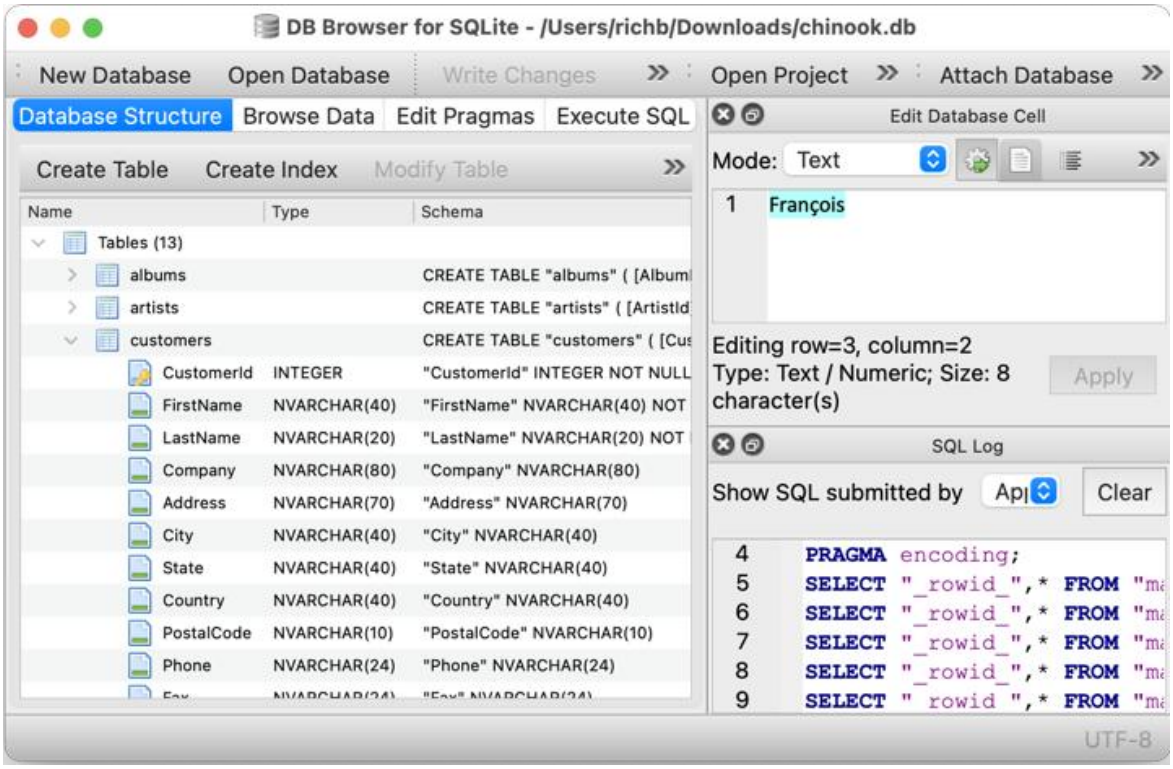
Framework je navíc snadno kombinovatelný s moderními frontend technologiemi, jako jsou React, Vue.js nebo Angular, což umožňuje vytvářet moderní a interaktivní uživatelská rozhraní. Z hlediska výkonu je výhodou i to, že data jsou přenášena přes požadavky a uchovávána v databázi nebo session, což znamená nižší nároky na RAM serveru a efektivnější chod celé aplikace.

1.2.6 Výber databázové technologie

Pro ukládání dat byla zvolena databázová technologie SQLite, a to především kvůli její jednoduchosti a nízkým nárokům na systémové prostředky. Jednou z hlavních výhod je, že nevyžaduje žádnou složitou konfiguraci – jedná se o integrovanou databázi, která funguje bez potřeby samostatného databázového serveru. Celá databáze je uložena v jednom souboru, což výrazně usnadňuje její správu a nasazení.

SQLite je navíc optimalizována pro rychlý zápis i čtení dat a je ideální pro menší až středně velké aplikace, které nepotřebují složitou databázovou infrastrukturu. Velkou výhodou je také její dobrá integrace s ASP.NET Core – pomocí Entity Framework Core je možné jednoduše propojit databázi s aplikací bez složitého nastavování.

Díky své jednoduchosti je SQLite velmi vhodná i pro vývoj a testování, protože umožňuje rychlé nasazení bez nutnosti instalovat a konfigurovat plnohodnotný databázový systém. Navíc je ideálním řešením pro aplikace, které mají fungovat i v offline režimu, protože nevyžaduje připojení k žádnému externímu databázovému serveru. [8]



Obrázek 1- SQL lite browser (zdroj: [9])

1.2.7 Výběr frontendové technologie

Pro návrh a implementaci uživatelského rozhraní byla zvolena kombinace technologií Razor Pages a Bootstrap, které společně umožnily vytvořit moderní, přehledné a responzivní rozhraní s důrazem na jednoduchost a použitelnost.

Razor Pages, jako součást frameworku ASP.NET Core, představuje moderní přístup k vývoji webového rozhraní, který efektivně propojuje HTML šablony s programovou logikou v jazyce C#. Tento model je velmi vhodný pro aplikace, které nevyžadují složité frontendové frameworky, ale zároveň kladou důraz na čistotu a přehlednost kódu. Výhodou Razor Pages je přirozené oddělení prezentační vrstvy od aplikační logiky, čímž se usnadňuje správa a údržba jednotlivých komponent. V rámci práce byl tento přístup použit zejména při zpracování formulářů, validaci vstupních dat a zobrazení výstupů v reálném čase. Vývojový model navíc podporuje silnou integraci s dalšími nástroji v rámci ASP.NET, jako je model binding, tag helpers nebo middleware.[10]

Pro vizuální stránku aplikace byl zvolen framework Bootstrap, který se stal standardem pro rychlý vývoj webového rozhraní díky předem připraveným stylům, komponentám a responzivnímu grid systému. Bootstrap výrazně urychlil proces návrhu a vývoje tím, že poskytl hotové šablony pro běžně používané prvky, jako jsou tlačítka, formulářová pole nebo rozbalovací nabídky. Díky využití tohoto frameworku bylo možné snadno zajistit konzistentní vzhled celé aplikace.[11]

Bootstrap také umožňuje jednoduché přizpůsobení aplikace pro různá zařízení – od velkých monitorů až po mobilní telefony – a tím zajišťuje dobrou použitelnost napříč platformami. V praxi byl použit například pro rozvržení navigačního menu, formulářů pro zadávání transakcí, filtračních prvků nebo karet pro zobrazení detailů. Framework navíc obsahuje sadu utilit pro zarovnání, barevné varianty, ovládání velikostí i viditelnosti prvků, což umožňuje rychle reagovat na potřeby designu bez zbytečných zásahů do kódu.[12]

Použitím těchto dvou technologií se podařilo vytvořit přehledné, funkční a vizuálně příjemné rozhraní. Kombinace Razor Pages a Bootstrapu je zároveň ideální pro menší až středně velké projekty, kde je důraz kladen na rychlost vývoje, srozumitelnost kódu a snadné rozšíření do budoucna.

1.2.8 Výběr vedlejších technologií

Kromě hlavních technologií použitých při vývoji backendové a frontendové části aplikace byly v projektu využity také doplňkové nástroje a knihovny, které rozšířily funkcionalitu systému a zlepšily jeho použitelnost z pohledu koncového uživatele. Tyto technologie nebyly součástí hlavního vývojového frameworku, ale významně přispěly k celkové kvalitě a přehlednosti aplikace.

Jednou z těchto technologií byla Nominatim, veřejně dostupná API služba postavená na projektu OpenStreetMap, která slouží k tzv. *geokódování* – tedy převodu textově zadané adresy na zeměpisné souřadnice (zeměpisná šířka a délka). Tato služba byla využita ve chvíli, kdy uživatel zadal adresu v rámci nové transakce. Pomocí HTTP požadavku byla adresa odeslána na API službu Nominatim, která následně vrátila odpověď ve formátu JSON s odpovídajícími souřadnicemi. Tyto údaje byly následně využity při vizualizaci dat v mapě a také při filtrování a třídění transakcí podle lokality. Výhodou Nominatimu je jeho otevřenost, snadná integrace a dostupnost bez nutnosti registrace či komerční licence. Ve vývoji tak bylo možné rychle nasadit geografické funkce bez nutnosti využívat složitější a nákladnější mapové služby.[13]

Další důležitou technologií použitou v této práci byla knihovna Chart.js, která slouží pro vizualizaci dat přímo v prostředí webového rozhraní. Chart.js je moderní JavaScriptová knihovna umožňující jednoduché a interaktivní vykreslování různých typů grafů – například sloupcových, koláčových nebo čárových. V aplikaci byla tato knihovna využita pro zobrazení přehledných grafů zobrazujících například rozdělení výdajů podle kategorií, poměr příjmů a výdajů nebo vývoj finanční bilance v čase. Díky využití Chart.js bylo možné nabídnout uživateli intuitivní a vizuálně přitažlivý způsob práce s daty bez nutnosti externích nástrojů či desktopového prostředí.[14]

Obě tyto technologie – Nominatim a Chart.js – významně přispěly k vyšší přidané hodnotě výsledné aplikace. Nominatim rozšířil aplikaci o možnosti zobrazování geografických dat v rámci aplikace, zatímco Chart.js umožnil rychlou vizualizaci dat, která by jinak byla zobrazena pouze v textové nebo tabulkové podobě. Jejich využitím se podařilo zkombinovat technickou efektivitu s uživatelskou přívětivostí, což odpovídá aktuálním trendům ve vývoji moderních webových systémů.

1.3 Fungování ASP.NET Core MVC aplikace

1.3.1 Komponenty ASP.NET Core MVC aplikace

Každá ASP.NET Core MVC aplikace je složena z několika základních komponent, které společně definují její chování a rozšiřitelnost. Mezi tyto komponenty patří:

Model-View-Controller (MVC)

Architektura Model-View-Controller (zkráceně MVC) je návrhový vzor, který rozděluje aplikaci do tří samostatných částí, čímž zvyšuje přehlednost, udržovatelnost a rozšiřitelnost kódu.[15; 6]

První částí je Model, který reprezentuje data a zajišťuje jejich správu a komunikaci s databází. Obsahuje obchodní logiku aplikace a definuje, jaká data se mají uchovávat a jak s nimi pracovat.

Druhou částí je View, která má na starosti zobrazování dat uživateli. Z pohledu uživatele představuje rozhraní, se kterým pracuje – zobrazuje tabulky, formuláře, texty a další vizuální prvky.

Poslední částí je Controller, který zpracovává požadavky uživatele (např. kliknutí na tlačítko nebo odeslání formuláře). Controller zajišťuje logiku zpracování, komunikuje s modelem a následně předává výsledná data zpět do view, nebo je odesílá prostřednictvím API.

Tento způsob rozdělení usnadňuje týmový vývoj, testování jednotlivých částí a celkově přispívá k lepší organizaci kódu v rozsáhlejších aplikacích.[16]

Middleware

ASP.NET využívá middleware, což jsou komponenty, které zpracovávají HTTP požadavky dříve než se dostanou ke kontrollerům. Pomocí middlewaru je možné přidat autentizaci, logování nebo i zpracovávání chyb.[17]

Routing

Routing se využivá k mapování URL adresy na konkretní akce a kontrolery. V ASP.NET se konfiguruje v souboru Startup.cs nebo program.cs. Routing nám umožnuje brát i parametry z URL například, když uživatel zadá do URL https://mujobchod.cz/produkty a přidá například /detail/7 tak to uživateli dovolí zobrazit si detailní informace o produktu s id 7 v našem obchodu.[18]

Razer view engine

Razor je šablonovací jazyk a zároveň view engine, který se používá v ASP.NET Core MVC k dynamickému generování HTML stránek na straně serveru. Umožňuje psát kombinaci C# kódu a HTML v rámci jednoho souboru.

Razor je navržen tak, aby byl jednoduchý na použití, čitelný a efektivní, přičemž eliminuje potřebu používat samostatné šablonovací jazyky jako PHP nebo JSP. [10]

Entity framework

Entity Framework (EF) je ORM (Object-Relational Mapper), který usnadňuje práci s databázemi v ASP.NET Core. Umožňuje vývojářům pracovat s databázovými tabulkami jako s objekty a používat C# místo SQL dotazů.

Entity Framework využívá Code first, což umožnuje nadefinovat databázi v C# třídách a poté z nich rovnou vytvoří samostatnou databázi. V tomto postupu je každá tabulka databáze reprezentována pomocí jedné třídy. [19]

```
public class Product
{
    Počet odkazů: 0
    public int Id { get; set; } // Primární klíč
    Počet odkazů: 0
    public string Name { get; set; } // Název produktu
    Počet odkazů: 0
    public decimal Price { get; set; } // Cena produktu
}
```

Obrázek 2 – C# třída pro Entity Framework (zdroj: vlastní)

1.3.2 Životní cyklus aplikace

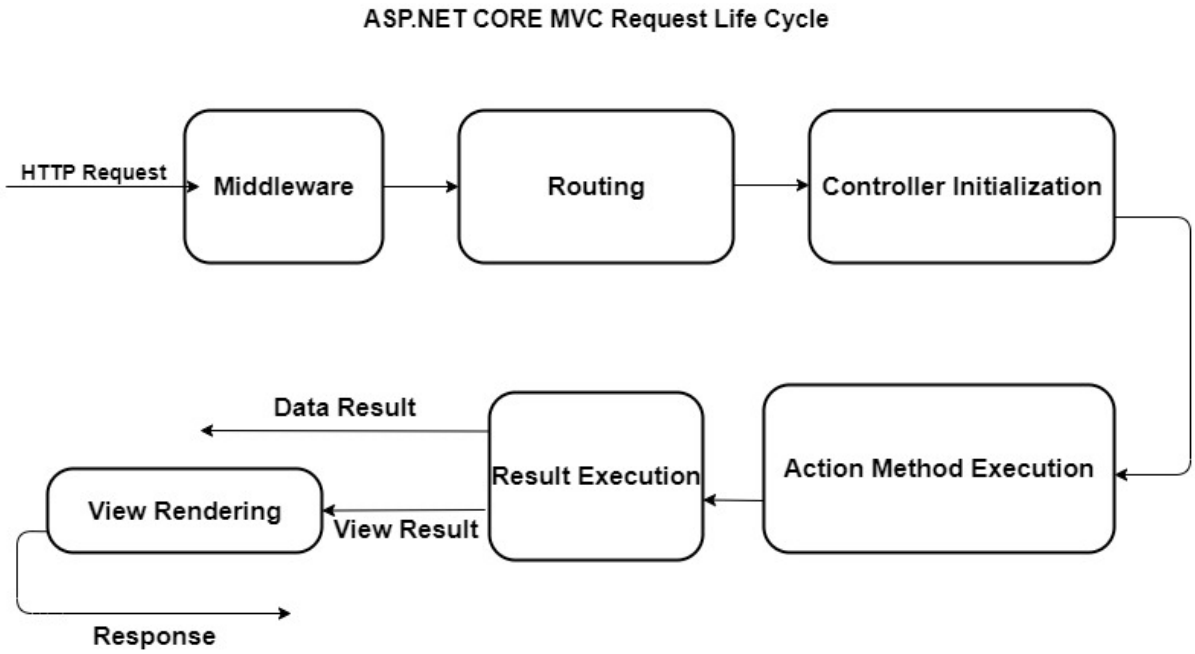
V ASP.NET Core MVC začíná životní cyklus požadavku přijetím HTTP žádosti webovým serverem, jako je například Kestrel nebo IIS. Tento server požadavek předá do middleware pipeline, která ho nadále zpracovává podle nakonfigurovaných komponent.[20]

Celý proces začíná tím, že klient odešle HTTP požadavek, který je přijat webovým serverem a předán aplikaci ASP.NET Core. Požadavek následně prochází middleware pipeline, což je sada komponent definovaných v souboru *Startup.cs*. Middleware může provádět různé operace, jako je autentizace a autorizace uživatele, logování, komprese odpovědi nebo manipulace s obsahem požadavku.

Dalším krokem je směrování (Routing), které rozhoduje o tom, jaký kontroler a jaká akční metoda budou zavolány na základě pravidel definovaných v aplikaci. Po identifikaci správné cesty framework vytvoří instanci odpovídající třídy kontroleru a zavolá její metodu, která zpracuje požadavek. Kontroler pak může komunikovat s modelovou vrstvou, načíst data z databáze nebo provést jinou potřebnou operaci.

Po zpracování kontroleru se vygeneruje odpověď v podobě ActionResult, což může být například ViewResult pro zobrazení HTML stránky, JsonResult pro vrácení JSON dat nebo RedirectResult pro přesměrování na jinou adresu. Nakonec middleware pipeline zajistí, že je odpověď správně serializována a odeslána zpět klientovi, a pokud se jedná o HTML obsah, zobrazí se v uživatelově prohlížeči.

Tento cyklus se opakuje při každém novém požadavku, což umožňuje dynamické generování obsahu na základě uživatelských interakcí. Výhodou ASP.NET Core MVC je jeho flexibilita a modularita, která umožňuje snadné přizpůsobení middleware komponent a rozšíření funkcionalit aplikace.



Obrázek 3 - Životní cyklus ASP.NET Core MVC Aplikace (zdroj: [20])

1.3.3 Testování webové aplikace aplikace

Unit testing (testování jednotlivých jednotek) je metodologie, která se zaměřuje na ověřování správnosti jednotlivých komponent aplikace, jako jsou funkce, metody nebo třídy. V kontextu webových aplikací, zejména těch postavených na frameworku jako ASP.NET Core MVC, hraje unit testing klíčovou roli v zajištění stability a kvality aplikace.

Princip Unit Testování

Unit testy se zaměřují na testování jednotlivých částí kódu izolovaně od zbytku aplikace. Cílem je ověřit, že každá komponenta funguje podle očekávání. Testy by měly být:

Izolované – testují pouze jednu část kódu (například jednu funkci nebo metodu) bez závislosti na ostatních.

Automatizované – mohou být spuštěny automaticky, což umožňuje pravidelnou kontrolu funkčnosti aplikace během vývoje.

Opakovatelné – testy by měly dávat stejné výsledky při každém spuštění, pokud se nezmění samotný kód, který testují.[21]

Význam Unit Testování pro webové aplikace

Unit testy jsou zásadní pro udržení kódu aplikace v dobré kondici, zejména při implementaci složitějších funkcí a nových funkcionalit. V případě webových aplikací, jsou unit testy nezbytné pro:

Zajištění správnosti kódu: Pomocí unit testů lze automaticky ověřit, že jednotlivé metody a funkce aplikace správně vykonávají požadované operace.

Rychlé odhalení chyb: Testy umožňují rychle detekovat regresní chyby, kdy po změně kódu dojde k porušení již fungujících částí aplikace.

Snazší údržbu a rozšíření: Díky testům je možné bezpečně přidávat nové funkce, protože změny v kódu mohou být okamžitě otestovány, aniž by bylo nutné ručně ověřovat všechny funkce aplikace.

Dokumentace chování aplikace: Unit testy slouží jako forma dokumentace, která ukazuje, jak jednotlivé části aplikace fungují, což je užitečné pro nové členy týmu nebo při předání projektu.

Testování v ASP.NET Core MVC

V ASP.NET Core MVC aplikacích je unit testování obzvláště efektivní díky podpoře pro unit testy přímo v rámci frameworku. ASP.NET Core využívá Dependency Injection, což znamená, že je možné snadno „injektovat“ závislosti (jako databázové kontexty nebo externí služby) do tříd, které se testují, což usnadňuje psaní izolovaných testů.

Pro testování lze využít nástroje jako xUnit, NUnit nebo MSTest, které jsou široce podporovány v ekosystému .NET. Testování tak zahrnuje:

Testování kontrolerů: Ověření, zda kontrolery správně reagují na HTTP požadavky a poskytují správné odpovědi.

Testování modelů: Validace, zda modely správně zpracovávají data a komunikují s databází.

Testování služeb a logiky: Otestování obchodní logiky, která je oddělena od kontrolerů.[22]

2 EXPERIMENTÁLNÍ ČÁST

2.1 Analýza existujících webových aplikací pro správu finance

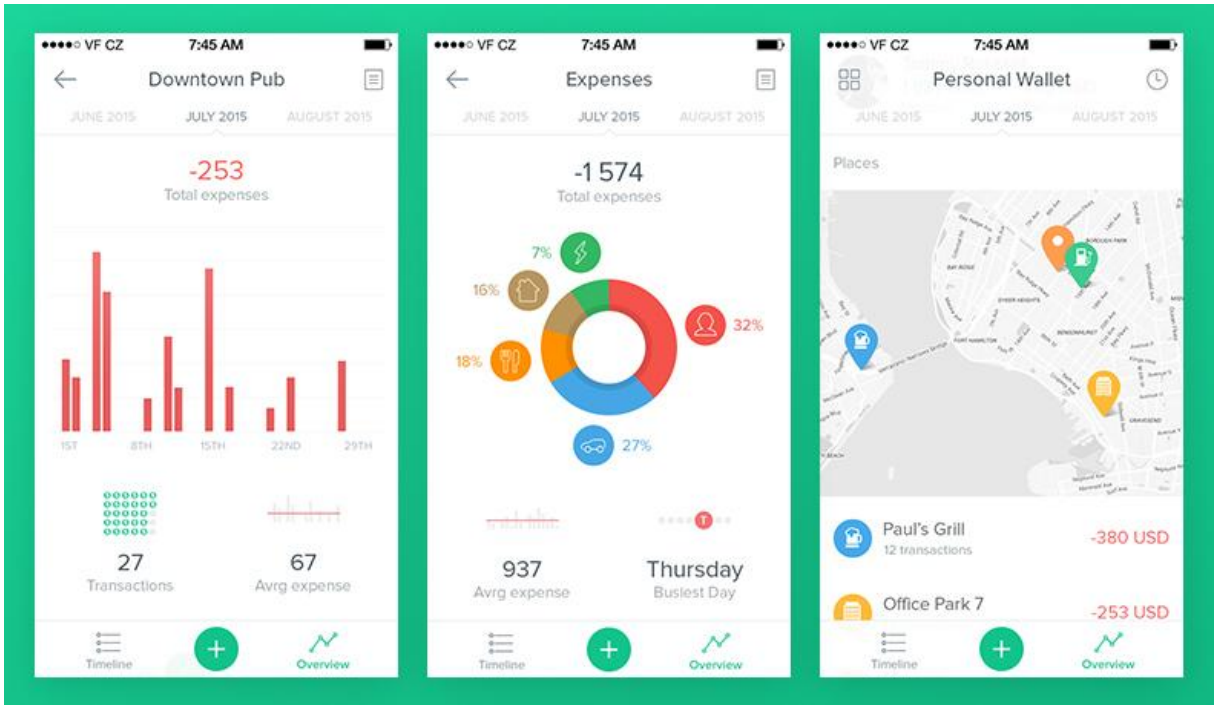
Na webu existuje řada aplikací zaměřených na správu osobních financí, které uživatelům umožňují sledovat příjmy, výdaje, investice a další finanční operace. V této části se zaměřím na analýzu nejpopulárnějších webových aplikací pro správu financí, které jsou k dispozici. Budu hodnotit jejich funkce, uživatelskou přívětivost, možnosti integrace s bankovními účty, připojení k API pro získávání finančních dat a také jejich možnosti vizualizace financí prostřednictvím grafů a tabulek. Tento přehled pomůže zjistit, jaké silné a slabé stránky tyto aplikace mají a jaké funkce jsou pro uživatele nejužitečnější.

2.1.1 Spendee

Spendee je finanční aplikace vytvořena v Česku. Je dostupná jak pro Android, IOS tak i pro web. Aplikace je dostupná od roku 2013 na IOS s tím že dnes jí využívá přes 3 milióny lidí na celém světě. Aplikace dovoluje ve zdarma plánu zobrazovat detailně osobní finance a import a export dat. Plus plán navíc dovoluje mít neomezený počet peněženek a sdílet peněženku s ostatními lidmi. Plus plán stojí 15$ na rok. Premium plán dovoluje vše předchozí a k tomu dovoluje automaticky kategorizovat transakce a umožnuje připojit bankovní účty podporovaných bank k automatickému přidávání transakcí. Premium plán stojí 23 $ na rok.

Mezi hlavní výhody patří dostupnost aplikace v české verzi, což usnadňuje její používání pro české uživatele. Aplikace je k dispozici nejen na zařízeních s iOS a Androidem, ale také ve webové verzi, což umožňuje pohodlný přístup odkudkoli. Nabízí zdarma předplatné, takže základní funkce jsou dostupné bez nutnosti platby. Uživatelé mají k dispozici detailní zobrazení svých osobních financí, což pomáhá s jejich efektivní správou. Důraz je kladen také na zabezpečení dat, aby byly osobní informace chráněny.

Mezi nevýhody patří to, že velká část užitečných funkcí je dostupná pouze v rámci placených plánů Premium a Plus, což může omezit možnosti uživatelů ve verzi zdarma. Aplikace může být pomalejší, což může ovlivnit plynulost jejího používání. Dalším problémem je nestabilní připojování bank – ne vždy se podaří propojení úspěšně dokončit, a to i v případě, že daná banka je oficiálně podporována.[23]



Obrázek 4 - Aplikace Spendee[24]

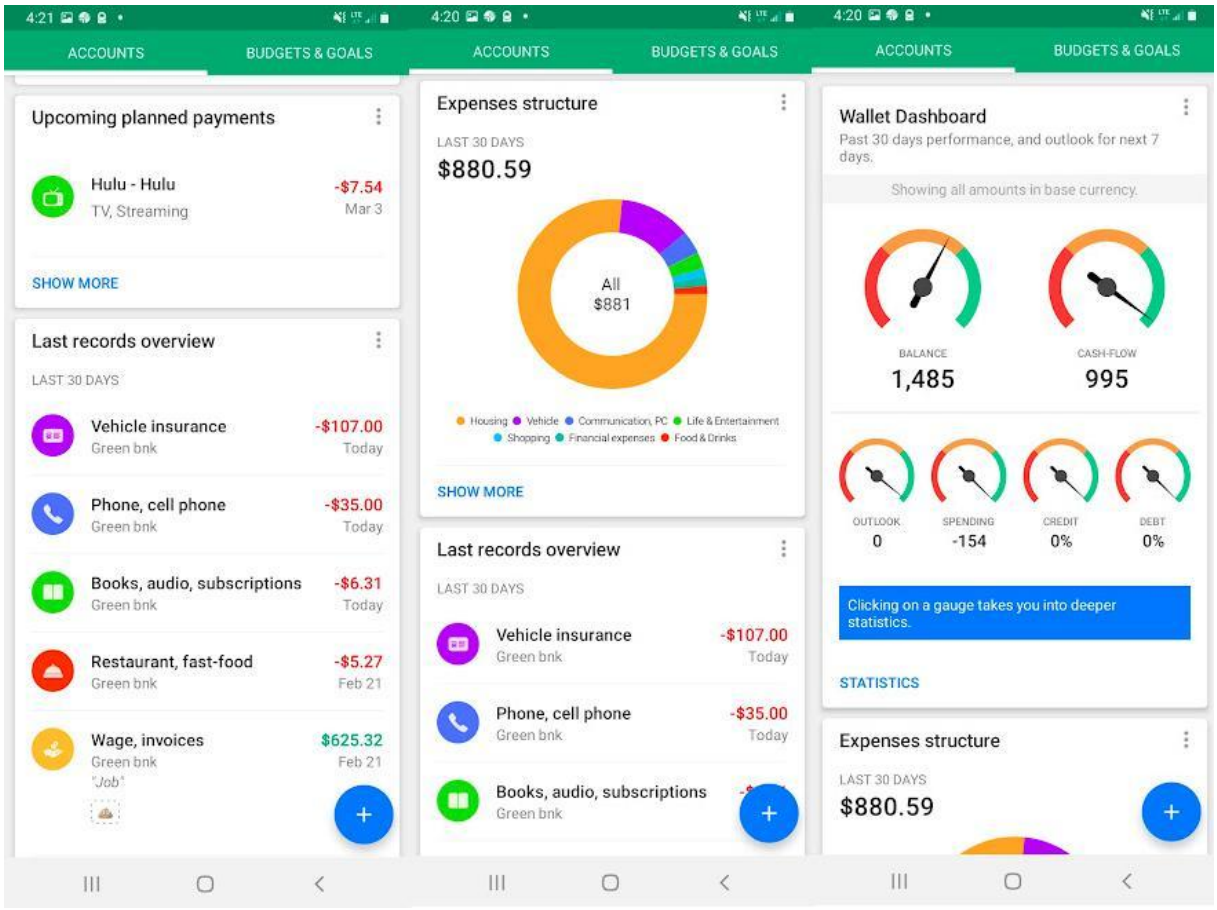
2.1.2 Wallet

Wallet je česká finanční aplikace, která je k dispozici pro Android, iOS i na webu. Aplikace byla vytvořena firmou BudgetBakers a je populární mezi uživateli, kteří chtějí snadno spravovat své osobní finance. Aplikace je dostupná od roku 2012 a dnes ji využívá více než 6 milionů uživatelů na celém světě. Wallet umožňuje připojení bankovních účtů pro automatický import transakcí, což usnadňuje správu financí. Kromě sledování příjmů a výdajů nabízí také detailní analýzu výdajů a příjmů, možnost nastavení rozpočtů a vizualizace dat ve formě grafů.

Wallet nabízí bezplatnou verzi, která umožňuje sledování výdajů a příjmů, propojení s bankovními účty, kategorizaci transakcí a generování přehledů. Placené verze, jako je Wallet Premium, umožňují pokročilejší funkce jako připojení více bankovních účtů, synchronizaci mezi více zařízeními, a nabídku pokročilých reportů a analýz. Cena Premium verze je 29,99 USD za rok.

Mezi hlavní výhody patří podpora pro více než 15 000 bank a finančních institucí. Uživatelé si mohou propojit více účtů a získat tak lepší přehled o svých financích. Aplikace nabízí detailní analýzu a vizualizaci výdajů, což pomáhá s efektivnějším řízením rozpočtu. Kromě toho umožňuje vytváření rozpočtů a finančních cílů, což uživatelům usnadňuje plánování. Další užitečnou funkcí je možnost sdílení peněženek s ostatními uživateli, například s rodinou nebo přáteli.

Mezi nevýhody patří to, že pokročilé funkce jsou dostupné pouze ve verzi Premium, což může omezit možnosti uživatelů ve verzi zdarma. Některé funkce jsou navíc omezené, pokud není účet připojen k aplikaci. Dalším nedostatkem je, že ne všechny banky jsou podporovány pro automatický import transakcí, což může vyžadovat ruční zadávání některých údajů.[25]



Obrázek 5 - Aplikace Wallet (zdroj: [25]

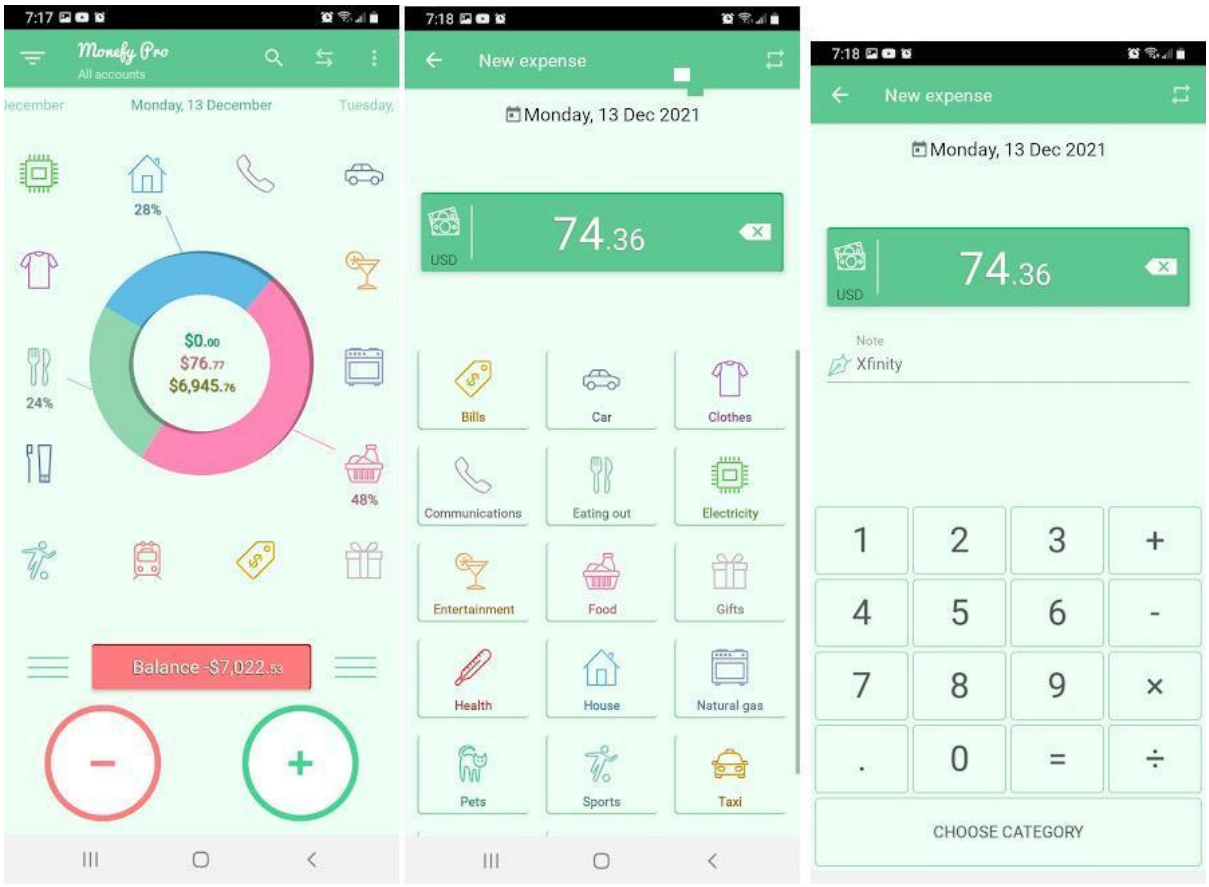
2.1.3 Monefy

Monefy je populární mobilní aplikace pro správu osobních financí, která je dostupná pro zařízení s Androidem i iOS. Aplikace je zaměřena na jednoduchost a uživatelskou přívětivost. Monefy umožňuje snadné sledování příjmů a výdajů, automatickou kategorizaci transakcí a podporuje propojení s bankovními účty. Uživatelé mohou sledovat své finance pomocí přehledných grafů, které vizualizují různé kategorie výdajů, a mohou snadno nastavit rozpočty pro jednotlivé kategorie. Aplikace nabízí základní verzi zdarma, která obsahuje většinu funkcí, ale pro pokročilejší funkce (např. synchronizaci mezi zařízeními) je potřeba zakoupit premium verzi.

Aplikace nabízí několik výhod, mezi které patří jednoduché rozhraní, které umožňuje snadné sledování financí bez složitého nastavování. Výdaje a příjmy jsou zobrazovány v přehledných grafech a analýzách, což usnadňuje kontrolu nad rozpočtem. Uživatelé prémiové verze mohou využít synchronizaci mezi zařízeními. Důraz je kladen také na bezpečnost, protože aplikace podporuje šifrování dat pro ochranu osobních údajů. Velkou výhodou je také rychlé přidávání transakcí, díky čemuž lze příjmy a výdaje snadno zaznamenávat během dne.

Mezi nevýhody patří omezení pokročilých funkcí pouze na Premium verzi, například synchronizace mezi zařízeními je dostupná jen pro platící uživatele. Aplikace nepodporuje automatické propojení s bankami, což znamená, že transakce je nutné zadávat ručně. Ve verzi zdarma jsou také některé funkce omezené, například detailní analýza výdajů podle kategorií není plně dostupná bez upgradu na Premium.

Monefy je dobrá volba pro uživatele, kteří preferují jednoduchost a snadné použití bez přílišného zbytečného rozhraní.[26]



Obrázek 6- Aplikace Monefy (zdroj: [26]

2.1.4 GoodBudget

GoodBudget je aplikace pro správu osobních financí, která je k dispozici na platformách Android,IOS a web. Je postavena na principu systému obálek, což znamená, že uživatelé mohou rozdělit své příjmy do různých "obálek" (kategorii výdajů) a sledovat, jak je používají. Aplikace je ideální pro ty, kteří preferují plánování a kontrolu svých výdajů na základě pevně stanoveného rozpočtu. GoodBudget je přístupná zdarma s možností zakoupení Premium verze, která přidává pokročilé funkce, jako je synchronizace mezi zařízeními a pokročilé reporty.

Aplikace nabízí několik výhod, které pomáhají uživatelům efektivně spravovat své finance. Uživatelé prémiové verze mohou využívat synchronizaci mezi více zařízeními, což zajišťuje pohodlný přístup k financím odkudkoli. Data jsou bezpečně uložena a lze je zálohovat, což minimalizuje riziko jejich ztráty. Aplikace umožňuje snadné zaznamenávání a sledování výdajů, přičemž je možné porovnávat skutečné výdaje s plánovaným rozpočtem. K lepší orientaci ve financích přispívají přehledné grafy, které vizualizují rozdělení peněz a jejich pohyb v různých kategoriích.

Mezi nevýhody patří absence bankovních propojení, což znamená, že aplikace neumožňuje automatické importování transakcí z bankovních účtů a všechny platby je nutné zadávat ručně. Některé pokročilé funkce, jako je synchronizace mezi zařízeními, detailní reporty nebo neomezený počet obálek, jsou dostupné pouze v placené verzi Premium. Aplikace je zaměřena na jednoduchost, což může být nevýhoda pro uživatele, kteří hledají pokročilejší nástroje pro správu financí.

GoodBudget je tak ideální pro ty, kteří chtějí využívat systém obálek k řízení rozpočtu a nevadí jim manuální zadávání transakcí.[27]

Obrázek 7 - Aplikace GoodBudget (zdroj: [27])

2.2 Návrh aplikace

Aplikace bude kompatibilní se všemi moderními webovými prohlížeči, včetně Chrome, Firefox, Safari a Edge . Pro správné fungování aplikace bude potřeba minimálně verze prohlížeče, která podporuje HTML5, CSS3 a JavaScript (například verze Chrome 60+, Firefox 60+, Safari 12+).

Aplikace bude nabízet základní funkce pro správu osobních financí, včetně sledování příjmů, výdajů, import dat, export dat a analýzy transakcí, aby uživatelům poskytla nástroje pro efektivní řízení jejich financí bez jakýchkoli zbytečných omezení.

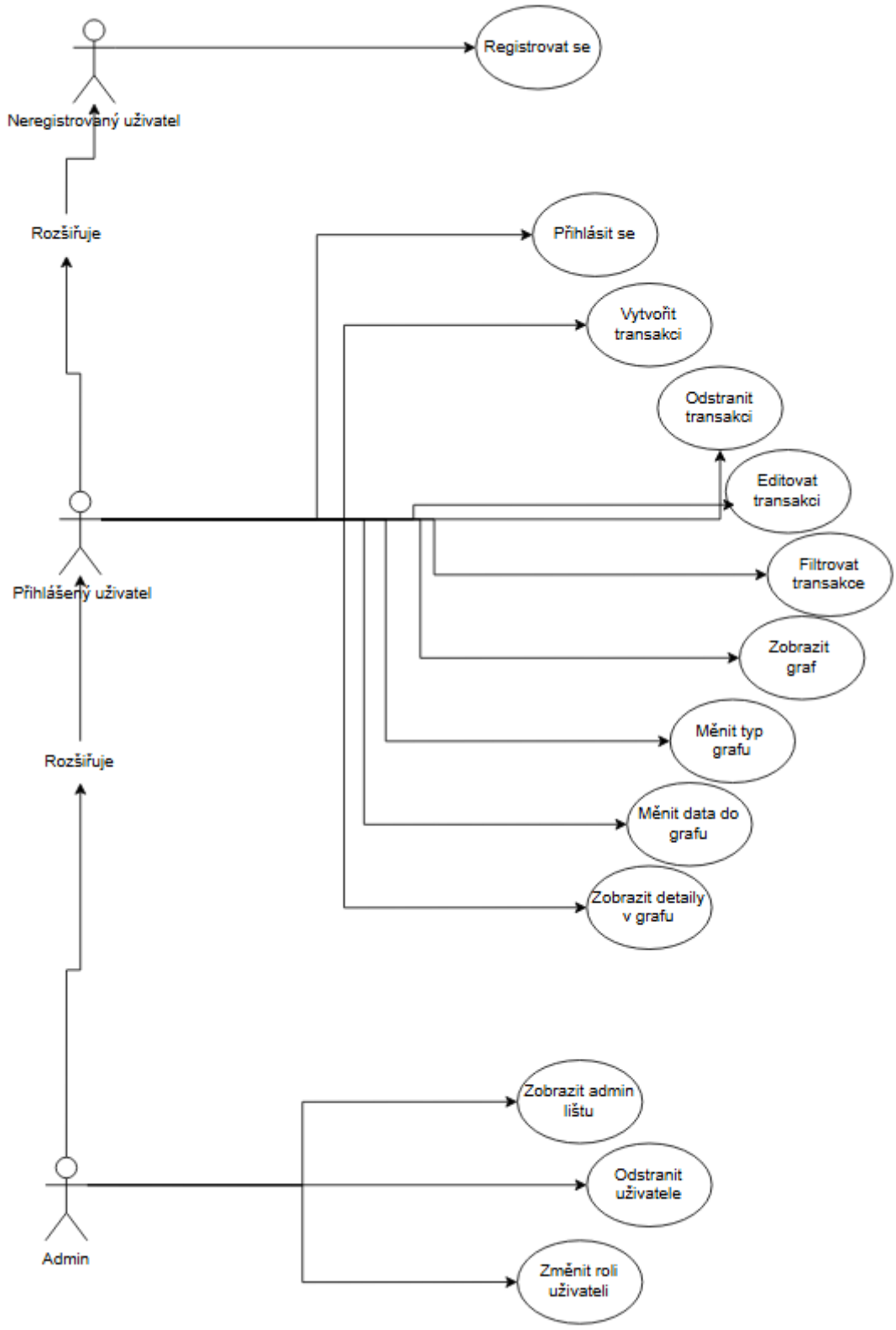
2.2.1 Požadavky systému

Funkční požadavky

- Uživatel může nahrávat data ve formátu CSV
- Uživatel může filtrovat v transakcích
- Uživatel může seřazovat transakce
- Uživatel může měnit, vytvářet a mazat transakce
- Uživatel má možnost si zobrazit všechny transakce v grafech
- Uživatel má možnost si filtrovat data pro jiné zobrazení v grafu
- Uživatel má možnost zobrazit si přesné transakce v grafu jako podkategorii
- Uživatel může měnit automaticky generovanou polohu transakce

Nefunkční požadavky

- Prostředí aplikace je v češtině
- Ochrana proti SQL Injection je zajištěna pomocí Entity Frameworku



Obrázek 8- Diagram případu užití (zdroj: vlastní)

2.2.2 Datový model

Tento datový model reprezentuje systém pro správu finančních transakcí, který obsahuje několik propojených tabulek. Hlavní tabulkou je Transactions, která uchovává informace o jednotlivých transakcích. Každá transakce má svůj unikátní identifikátor a obsahuje údaje jako číslo účtu, částku, bankovní kód, IBAN, SWIFT a datum transakce. Dále jsou zde cizí klíče odkazující na další tabulky, jako je identifikátor uživatele, měny, kategorie transakce, lokace a typ transakce.

Tabulka Users obsahuje informace o uživatelích systému. Každý uživatel má své unikátní uživatelské jméno, hash hesla a roli.

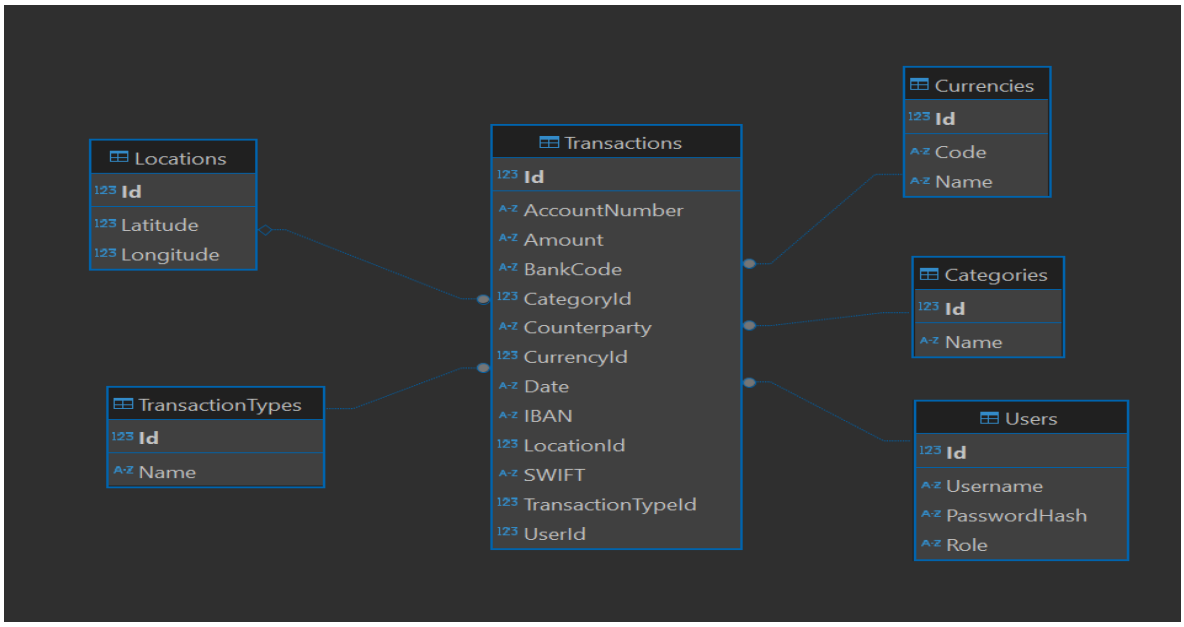
Tabulka Currencies slouží k uchování informací o měnách, které se v transakcích používají. Každá měna je identifikována unikátním kódem a názvem.

Tabulka Categories umožňuje rozdělovat transakce do různých kategorií, což může pomoci při jejich správě a analýze.

Tabulka TransactionTypes obsahuje různé typy transakcí, což umožňuje jejich klasifikaci na základě předem definovaných pravidel.

Tabulka Locations uchovává geografické údaje o místech, kde byly transakce provedeny. Každá lokalita je určena zeměpisnou šířkou a délkou, což může být užitečné například pro analytické účely.

Celý model je postaven na vzájemně propojených tabulkách, které umožňují efektivní správu transakcí, jejich kategorizaci, sledování lokací a zabezpečenou autentizaci uživatelů.



Obrázek 9 - ER Diagram (zdroj: vlastní)

2.2.3 Tvorba aplikace

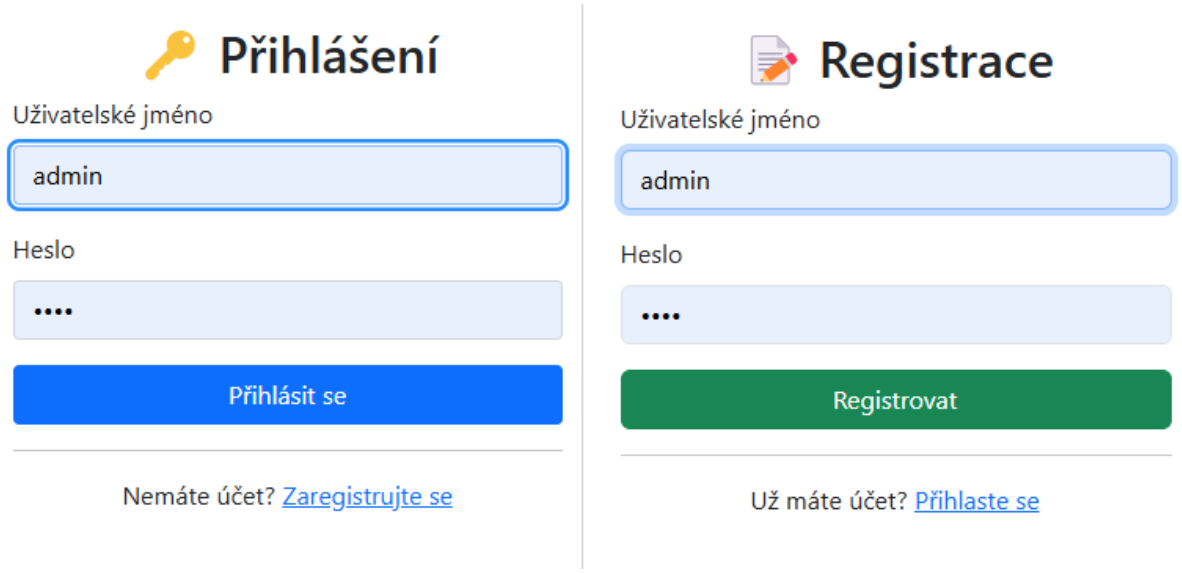
Backend Aplikace je napsána pomocí jazyka C#. Pro tvorbu bylo využito prostředí Visual Studio. Frontend část aplikace je napsána pomocí HTML, CSS, Bootstrapu a Javascriptu.

2.2.4 Uživatelské prostředí

Aplikace je rozdělena do pěti hlavních částí. První část se stará o přihlašování a registraci uživatelů. Druhá část se stará o zobrazení všech transakcí, jejich filtrování a šeřazování. Další část umožňuje spravovat každou transakci a upravovat její atributy. Čtvrtou částí aplikace je záložka Grafy, která umožnuje uživateli zobrazit několik typů grafů s tím, že zde má možnost filtrovat data v grafu podle datumů a typů transakcí. Poslední stránka je zobrazena pouze pro uživatele s rolí admin. Na té mohou admini spravovat uživatele.

Stránky přihlašování a registrace.

Zde se uživatel může přihlásit pomocí svého uživatelského jména a hesla. V případě, že nemá ještě své vlastní přihlašovací údaje, může si je vytvořit na stránce registrace.



Obrázek 10- Přihlášení a Registrace (zdroj: vlastní)

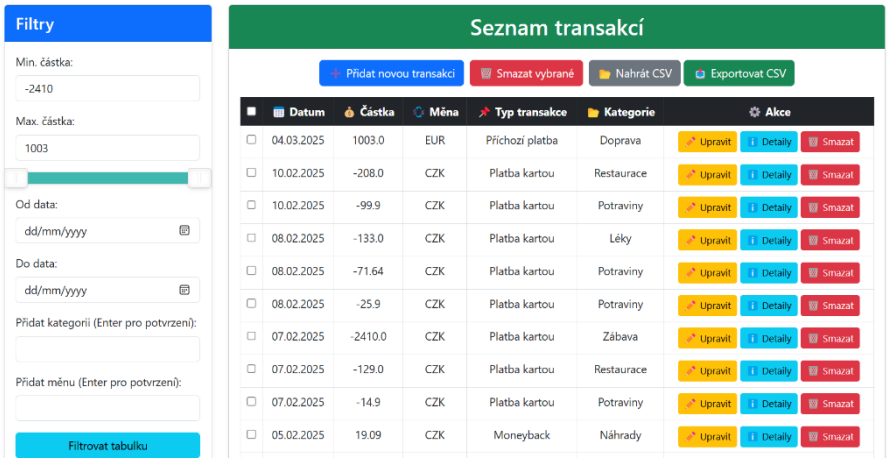
Stránka transakcí

Stránka transakcí se uživateli zobrazí ihned po přihlášení do aplikace. Na této stránce má uživatel přehled o všech transakcích, které dosud do aplikace vložil.

V levé části stránky se nachází panel pro filtrování transakcí podle data, kategorie, měny a rozmezí částky. Ve střední části obrazovky je hlavní seznam všech transakcí, který lze dle potřeby filtrovat a řadit. Hlavička seznamu umožňuje řazení transakcí podle data, měny, částky, kategorie nebo typu transakce.

Každou jednotlivou transakci může uživatel detailně zobrazit nebo upravit. Pokud je transakce chybná nebo ji uživatel v aplikaci dále nechce uchovávat, lze ji jednoduše odstranit.

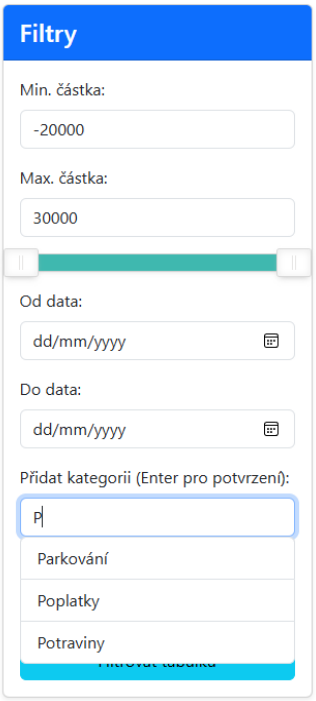
Nad hlavičkou seznamu jsou umístěna čtyři hlavní tlačítka, která usnadňují práci s transakcemi. První z nich umožňuje přidat novou transakci manuálním vložením, přičemž se ihned zobrazí v seznamu. Další tlačítko slouží k odstranění vybraných transakcí, což umožňuje snadné mazání nepotřebných záznamů. Funkce importu dat z CSV umožňuje pohodlně nahrát transakce ze souboru, čímž se eliminuje nutnost jejich ručního zadávání. Poslední tlačítko slouží k exportu transakcí do CSV souboru, což umožňuje jejich další zpracování v jiné aplikaci.



Obrázek 11 - Obrazovka Seznamu Transakcí (zdroj: vlastní)

Filtrovací mechanismus aplikace zahrnuje dvojitý posuvník (slider), který umožňuje uživatelům zadávat částku jak manuálním zadáním, tak prostřednictvím přetahování v definovaném rozmezí. Dále je implementována možnost výběru časového období pomocí komponenty pro výběr data (DatePicker), která umožňuje zadání počátečního a koncového data. Po zadání konkrétního data mohou uživatelé v sekcích kategorií a měn zadávat text,

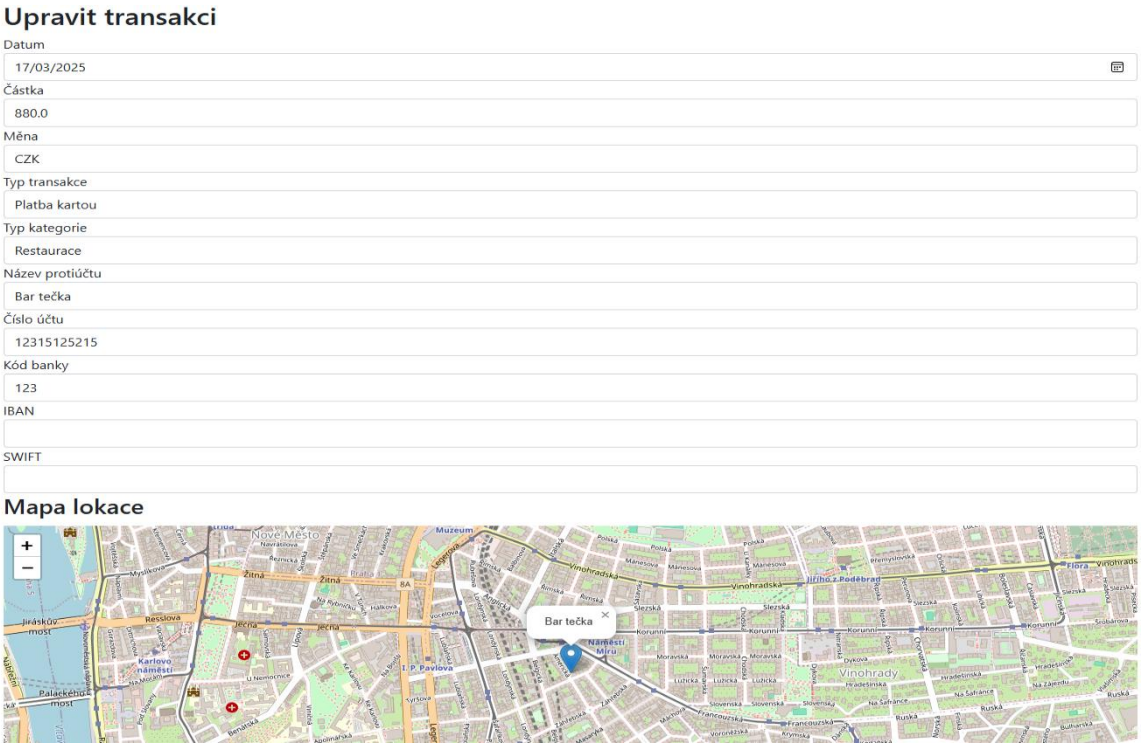
přičemž aplikace dynamicky nabízí relevantní možnosti filtrů prostřednictvím interaktivního výběrového pole.



Obrázek 12 - Filtr hlavní stránky (zdroj: vlastní)

Stránka editace

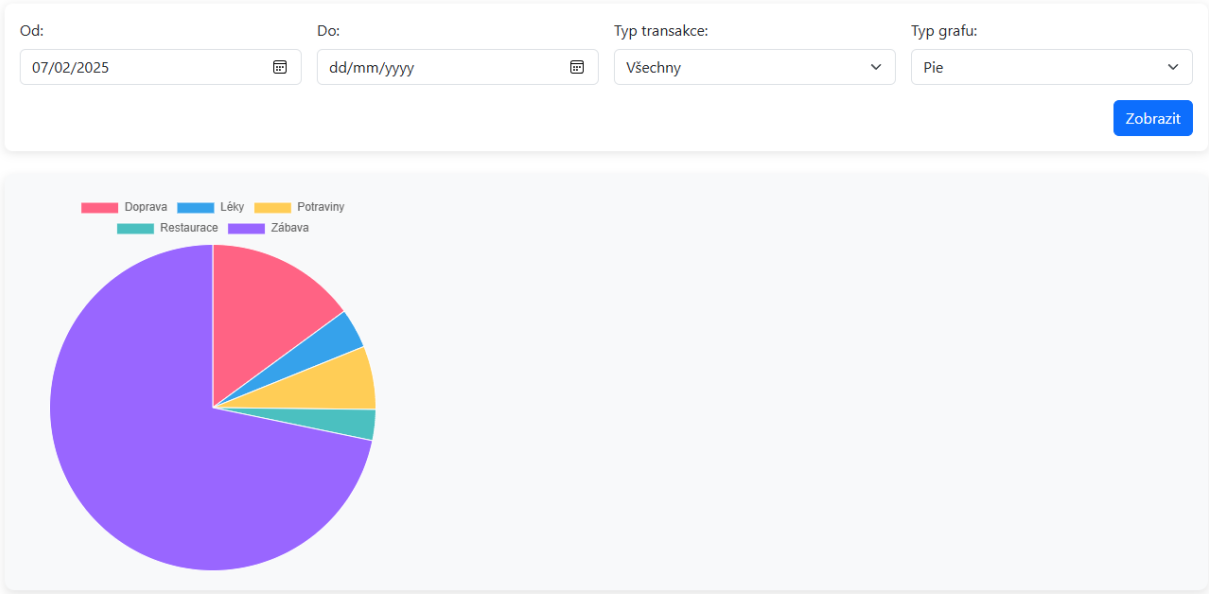
Stránka určená k editaci již vytvořených transakcí nabízí uživatelům detailní možnosti úprav. Každý atribut transakce může být libovolně změněn – ať už jde o datum, částku, kategorii, měnu či jakékoli další informace. Dále je na této stránce k dispozici možnost změnit polohu dané transakce v seznamu. Po dokončení všech požadovaných úprav a následném uložení se provedené změny ihned projeví v hlavním seznamu transakcí, což uživateli umožní okamžitou kontrolu nad provedenými změnami.



Obrázek 13 - Stránka editace transakce (zdroj: vlastní)

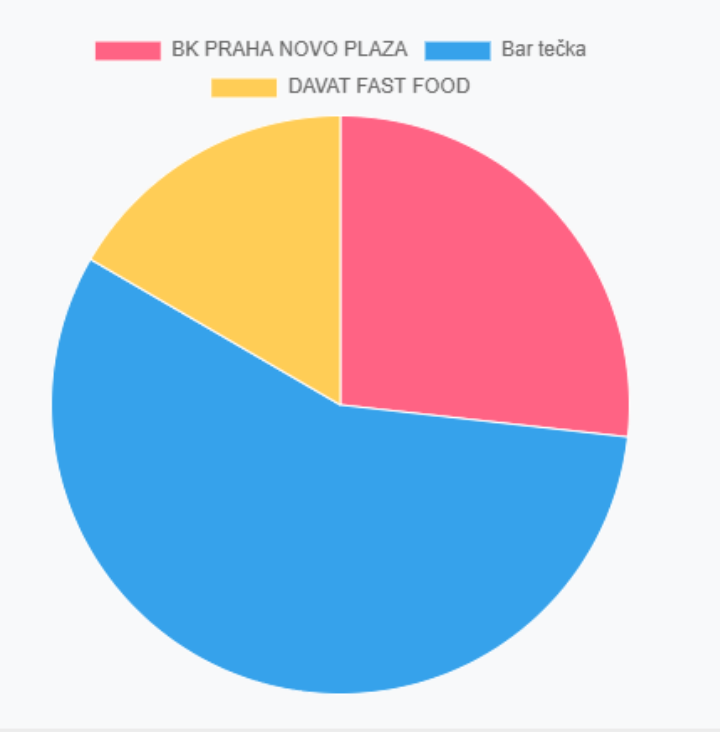
Stránka grafů

Obrazovka pro grafické zobrazení dat nabízí uživateli přehledné vizualizace transakcí prostřednictvím interaktivních grafů. Uživatel si může data v grafech upravit dle vlastních potřeb pomocí různých filtrů, jako jsou typy transakcí, konkrétní časové období a jednotlivé kategorie. Po nastavení filtrů si uživatel může vybrat ze čtyř typů grafů: spojnicový (Line), koláčový (Pie), sloupcový (Bar) a prstencový (Doughnut).



Obrázek 14 - Koláčový graf (zdroj: vlastní)

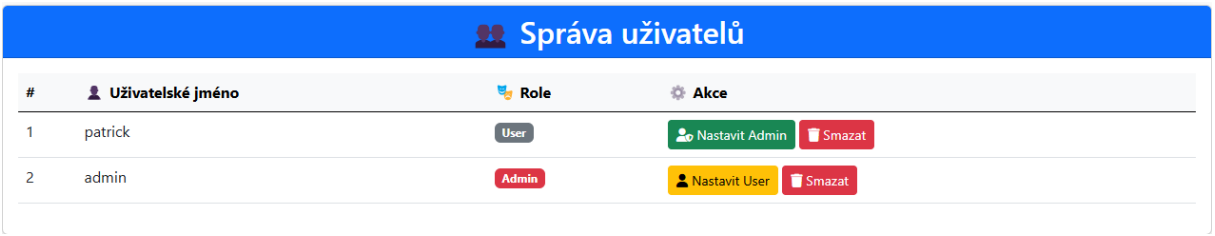
Všechny typy grafů navíc umožňují detailnější rozkliknutí podle jednotlivých kategorií, čímž uživatel získává přehled o konkrétních transakcích tvořících každou kategorii.



Obrázek 15 - Koláčový graf detail kategorie (zdroj: vlastní)

Stránka admin panelu

Tato stránka je dostupná pouze uživatelům, kteří mají přiřazenou roli administrátora. Umožňuje správu uživatelů v systému – konkrétně může administrátor měnit role ostatním uživatelům, například povýšit běžného uživatele na správce nebo naopak odebrat oprávnění. Kromě toho má také možnost uživatele zcela odstranit ze systému, pokud je to nutné. Tímto způsobem má administrátor kontrolu nad uživatelskou základnou a správou přístupových práv.



Obrázek 16- Admin panel (zdroj : vlastní)

2.2.5 Kód

Metoda GetLocationCoordinates slouží k získání zeměpisných souřadnic (zeměpisné šířky a délky) na základě textové adresy zadané uživatelem. Tato funkce využívá veřejné API služby Nominatim od OpenStreetMap, které umožňuje převod textové adresy na souřadnice (tzv. geokódování).

Na začátku metoda ověří, zda zadaná adresa není prázdná či neplatná. Pokud je vstup validní, vytvoří se instance HttpClient, pomocí které je proveden HTTP GET požadavek na rozhraní Nominatim API. V hlavičce požadavku je nastaven vlastní User-Agent, jak vyžaduje API.

Adresa se přidá do URL ve správně zakódovaném tvaru. Po odeslání požadavku metoda vyhodnocuje odpověď – pokud API vrátí úspěšný stav, načte se obsah odpovědi jako text a deserializuje se do objektu typu List<NominatimResult>. Tento objekt reprezentuje seznam nalezených výsledků, z nichž je použit první záznam.

Pokud se podaří úspěšně získat data, metoda vrací dvojici lat a lon (tedy zeměpisná šířka a délka). V opačném případě vrací null. Celý proces probíhá asynchronně, aby neblokoval běh aplikace, a je obalen v konstrukci try-catch, která zachytává případné výjimky při volání API nebo zpracování odpovědi.

```
private async Task<(double lat, double lon)?> GetLocationCoordinates(string address)
{
    if (string.IsNullOrWhiteSpace(address)) return null;

    using (HttpClient client = new HttpClient())
    {
        client.DefaultRequestHeaders.Add("User-Agent", "BakalarskaPraceApp/1.0 (your-email@example.com)");

        string url = $"https://nominatim.openstreetmap.org/search?format=json&q={Uri.EscapeDataString(address)}";
        Console.WriteLine($"API REQUEST: {url}");

        try
        {
            var response = await client.GetAsync(url);
            if (!response.IsSuccessStatusCode)
            {
                Console.WriteLine($"Chyba při volání API: {response.StatusCode}");
                return null;
            }

            var responseData = await response.Content.ReadAsStringAsync();
            var data = System.Text.Json.JsonSerializer.Deserialize<List<NominatimResult>>(responseData);

            if (data != null && data.Count > 0)
            {
                Console.WriteLine($"API odpověď: {data[0].lat}, {data[0].lon}");
                return (double.Parse(data[0].lat), double.Parse(data[0].lon));
            }
            else
            {
                Console.WriteLine($"API nenašlo žádné souřadnice pro {address}");
                return null;
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Chyba při volání API: {ex.Message}");
            return null;
        }
    }
}
```

Obrázek 17 - Metoda GetLocationCoordinates (zdroj: vlastní)

ZÁVĚR

Závěrem lze shrnout, že cílem této bakalářské práce bylo navrhnout a implementovat webovou aplikaci zaměřenou na efektivní správu osobních financí. Aplikace uživatelům umožňuje přehlednou správu transakcí, včetně jejich přidávání, úprav, filtrování a odstraňování. Významnou funkcí aplikace je možnost vizualizace dat prostřednictvím interaktivních grafů (spojnicový, koláčový, sloupcový a prstencový), které lze přizpůsobit podle uživatelových preferencí a potřeb.

V rámci teoretické části byly detailně popsány používané technologie a přístupy k vývoji webových aplikací, zejména ASP.NET Core MVC a databázový systém SQLite. Praktická část pak zahrnovala samotnou realizaci aplikace.

Výsledná aplikace splňuje hlavní požadavky zadání, tedy intuitivní uživatelské rozhraní, bezpečné ukládání dat a jednoduché zpracování finančních informací. Díky použití moderních technologií je aplikace škálovatelná, snadno udržovatelná a uživatelsky přívětivá. Do budoucna je možné aplikaci dále rozšířit například o automatickou integraci bankovních účtů nebo další analytické funkce pro pokročilou správu osobních financí.

POUŽITÁ LITERATURA

Reference

[1] *Should I Build a Desktop or Web Application?* [online]. 2024 [cit. 2025-03-31]. Dostupné z: https://medium.com/@GuruTechnolabs/should-i-build-a-desktop-or-web-application-b83deb81baf6

[2] *World Wide Web (WWW)* [online]. 2023 [cit. 2025-03-30]. Dostupné z: https://www.techtarget.com/whatis/definition/World-Wide-Web

[3] *How Does a Website Work? A Complete Guide for Beginners and Experts* [online]. 2024 [cit. 2025-03-30]. Dostupné z: https://www.bluehost.com/blog/how-websites-work/

[4] FELKE-MORRIS, Terry. *Web development and design foundations with HTML5*. 9th edition. New York, NY: Pearson, [2019]. ISBN 978-0-13-480114-8.

[5] *Static vs. dynamic websites: the key differences and which to use* [online]. 2024 [cit. 2025-03-29]. Dostupné z: https://www.wix.com/blog/static-vs-dynamic-website

[6] *MVC Framework Introduction* [online]. 2024 [cit. 2025-03-30]. Dostupné z: https://www.geeksforgeeks.org/mvc-framework-introduction/

[7] *Microsoft SQL Server* [online]. 2025 [cit. 2025-03-30]. Dostupné z: https://www.microsoft.com/en-us/sql-server

[8] *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems* [online]. 2022 [cit. 2025-03-29]. Dostupné z: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

[9] *DB Browser for SQLite* [online]. 2025 [cit. 2025-03-25]. Dostupné z: https://sqlitebrowser.org/

[10] *Introduction to Razor Pages in ASP.NET Core* [online]. 2024 [cit. 2025-03-30]. Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-9.0&tabs=visual-studio

[11] *Get started with Bootstrap* [online]. 2024 [cit. 2025-03-30]. Dostupné z: https://getbootstrap.com/docs/5.3/getting-started/introduction/

[12] *The pros and cons of using Bootstrap for front-end development* [online]. [cit. 2025-03-30]. Dostupné z: https://owdt.com/article/the-pros-and-cons-of-using-bootstrap-for-front-end-development/

[13] *Search queries* [online]. 2024 [cit. 2025-03-30]. Dostupné z: https://nominatim.org/release-docs/develop/api/Search/

[14] *Chart.js* [online]. 2025 [cit. 2025-03-30]. Dostupné z: https://www.chartjs.org/docs/latest/

[15] *ASP.NET MVC* [online]. 2024 [cit. 2025-03-25]. Dostupné z: https://www.h2kinfosys.com/blog/asp-net-mvc/

[16] FREEMAN, Adam. *Pro ASP.NET Core MVC 2*. Seventh edition. London: Apress, [2017]. ISBN 978-1-4842-3149-4.

[17] *ASP.NET Core Middleware* [online]. 2024 [cit. 2025-03-30]. Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-9.0

[18] *Routing in ASP.NET Core* [online]. 2024 [cit. 2025-03-30]. Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-9.0

[19] *Getting Started with EF Core* [online]. 2023 [cit. 2025-03-25]. Dostupné z: https://learn.microsoft.com/en-us/ef/core/get-started/overview/first-app?tabs=netcore-cli

[20] *ASP.NET Core MVC Request Life Cycle* [online]. 2024 [cit. 2025-03-30]. Dostupné z: https://www.csharp.com/article/asp-net-core-mvc-request-life-cycle/

[21] *What Is Unit Testing?* [online]. 2024 [cit. 2025-03-30]. Dostupné z: https://smartbear.com/learn/automated-testing/what-is-unit-testing/

[22] *Test ASP.NET Core MVC apps* [online]. 2022 [cit. 2025-03-30]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/test-asp-net-core-mvc-apps

[23] *Česká aplikace pro správu financí Spendee nově podporuje propojení s bankovními účty* [online]. 2017 [cit. 2025-03-24]. Dostupné z: https://cc.cz/ceska-aplikace-pro-spravu-financi-spendee-nove-podporuje-propojeni-s-bankovnimi-ucty/

[24] *Income vs. Spending on iPhone Reports!* [online]. 2015 [cit. 2025-03-24]. Dostupné z: https://goodbudget.com/blog/2015/10/income-vs-spending-on-iphone-reports/

[25] *Wallet by Budgetbakers: A Product Review* [online]. 2O22-06-11 [cit. 2025-03-24]. Dostupné z: https://incubatingwallet.com/wallet-by-budgetbakers-a-product-review/

[26] INCUBATING WALLET. *Monefy: A product Review* [online]. 2022 [cit. 2025-03-24]. Dostupné z: https://incubatingwallet.com/the-checkbook-app-monefy-product-review/

[27] *Goodbudget Review: A Hands-On Digital Envelope System* [online]. 2025 [cit. 2025-03-29]. Dostupné z: https://www.nerdwallet.com/article/finance/goodbudget-app-review