UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2024 Jakub Kodytek

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Aplikace pro sledování osobních výdajů a investic

Jakub Kodytek

Bakalářská práce

2024

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub Kodytek**
Osobní číslo:	**I21172**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Aplikace pro sledování osobních výdajů a investic**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Tato bakalářská práce se zaměřuje na vývoj desktopové aplikace pro sledování osobních výdajů a investic, zobrazování aktuálního stavu investičního trhu a následné grafické zobrazení uživatelských výdajů pro vybrané období. Cílem aplikace bude navrhnout aplikaci pro ukládání uživatelských výdajů v rámci různých kategorií a následné přehledné zobrazení v grafech pro určité období (měsíční/kvartální/roční). Aplikace bude zaměřená i na sledování investic.
Data pro investiční část budou získávána z webových API, přeformátována a pomocí periodického obnovování do aplikace zobrazována. Uživatelská data budou ukládána do databáze. Aplikace bude realizována v programovacím jazyku C# s využitím buď klasického .NET UI, nebo s frameworkem WPF, dále možné využití balíčků z balíčkovače Nugget.

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

Seznam doporučené literatury:

SELLS, Chris. *C# a WinForms: programování formulářů Windows*. Brno: Zoner Press, c2005. Encyklopedie Zoner Press. ISBN 80-86815-25-0.
SYROVÝ, Petr. *Investování pro začátečníky*. 4., zcela přepracované a rozšířené vydání. Praha: Grada Publishing, 2022. ISBN 978-80-271-3458-8.

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

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

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

V Pardubicích dne 28. února 2024

Prohlašuji:

Tuto bakalářskou 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., 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 5. 5. 2024

Jakub Kodytek

Poděkování

Rád bych poděkoval vedoucímu své bakalářské práce Ing. Janu Panušovi, Ph.D. za jeho ochotu při konzultacích, rady ke zpracování aplikace a odborné vedení. Dále mému kamarádovi, testerovi multiplatformních aplikací Bc. Tomáši Češkovi za zpětnou vazbu na návrh a řešení aplikace a v neposlední řadě mé rodině a kamarádům za podporu a motivaci.

Anotace

Bakalářská práce se zaměřuje na vývoj desktopové aplikace pro sledování osobních výdajů a investic, zobrazování aktuálního investičního trhu a následné grafické zobrazení uživatelských výdajů pro vybrané období. Cílem bakalářské práce bude navrhnout aplikaci pro ukládání uživatelských výdajů v rámci různých kategorií a následné přehledné zobrazení v grafech pro určité období – měsíční/kvartální/roční. Aplikace bude zaměřena i na sledování investic.

Klíčová slova

Finance, kryptoměny, WPF, API, C#, SQLite, grafy, databáze

Title

Application for tracking personal expenses and investments

Annotation

The bachelor's thesis focuses on the development of a desktop application for tracking personal expenses and investments, displaying the current investment market, and subsequently graphically presenting user expenses for a selected period. The goal of the bachelor's thesis will be to design an application for storing user expenses across various categories and their clear presentation in graphs for a certain period – monthly/quarterly/annually. The application will also be aimed at monitoring investments.

Keywords

Finances, cryptocurrencies, WPF, API, C#, SQLite, graphs, database

Obsah

Seznam ilustrací 10

Seznam zkratek a značek 11

Terminologie 12

Úvod 13

1 Analýza požadavků 14

1.1 Funkční požadavky 14

1.2 Nefunkční požadavky 15

2 Teoretický úvod 16

2.1 Kryptoměna 16

2.1.1 Bitcoin 16

2.2 Asymetrická kryptografie 16

2.3 Token 17

2.4 Blockchain 17

2.5 Tržní pojmy 18

3 Struktura aplikace 19

3.1 Moduly aplikace 19

3.1.1 Profil 19

3.1.2 Finance 20

3.1.3 Převodník 23

3.1.4 Kryptoměny 25

3.1.5 Dashboard 27

3.2 Kód 31

3.2.1 Struktura kódu 31

3.2.2 Singleton 33

4 Technická dokumentace 34

4.1 Použité technologie 34
4.1.1 HTTP komunikace 34
4.1.2 Asynchronizace 34
4.1.3 Jazyk C# 35
4.1.4 SQLite 35
4.1.5 API 35
4.1.6 WPF 36
4.1.7 .NET 36
4.1.8 XAML 37
4.2 Knihovny 38
4.2.1 LiveChartsWPF 38
4.2.2 Newtonsoft.Json 38
4.2.3 System.Data.SQLite 39
4.2.4 LINQ 39
4.2.5 CultureInfo 39
4.3 Databázová vrstva aplikace 40
4.3.1 Vkládání dat do databáze 40
4.3.2 Relační model 41
4.3.3 Popis tabulek 41
5 Budoucí rozšíření 42
5.1 Profil 42
5.2 Finance 42
5.3 Převodník 42
5.4 Kryptoměny 43
5.5 Dashboard 43
Závěr 44
Použitá literatura 45

Seznam ilustrací

Obrázek 1: Modul profil [zdroj vlastní] 19
Obrázek 2: Modul Finance – záložka nové [zdroj vlastní] 20
Obrázek 3: Modul Finance – validace částky [zdroj vlastní] 20
Obrázek 4: Modul Finance – validace názvu kategorie [zdroj vlastní] 21
Obrázek 5: Modul Finance – validace datumu [zdroj vlastní] 21
Obrázek 6: Modul finance – záložka evidované [zdroj vlastní] 22
Obrázek 7: Modul Převodník – ukázka naplněného combo boxu [zdroj vlastní] 23
Obrázek 8: Modul Převodník – validace hodnoty [zdroj vlastní] 23
Obrázek 9: Modul Převodník – ukázka úspěšného převodu [zdroj vlastní] 24
Obrázek 10: Modul Kryptoměny – záložka obecné [zdroj vlastní] 25
Obrázek 11: Modul Kryptoměny – záložka osobní [zdroj vlastní] 26
Obrázek 12: Modul Dashboard – ukázka dialogu při chybném výběru dat [zdroj vlastní] 27
Obrázek 13: Modul Dashboard – ukázka legendy grafu [zdroj vlastní] 28
Obrázek 14: Modul Dashboard – měsíční přehled, koláčový graf [zdroj vlastní] 28
Obrázek 15: Modul Dashboard – kvartální přehled, koláčový graf [zdroj vlastní] 29
Obrázek 16: Modul Dashboard – roční přehled, sloupcový graf [zdroj vlastní] 30
Obrázek 17: Struktura kódu – balíček export [zdroj vlastní] 31
Obrázek 18: Struktura kódu – balíček graphics [zdroj vlastní] 31
Obrázek 19: Struktura kódu – balíček model [zdroj vlastní] 32
Obrázek 20: Struktura kódu – balíček utility [zdroj vlastní] 32
Obrázek 21: Struktura kódu – návrhový vzor Singleton [zdroj vlastní] 33
Obrázek 22: Příklad XAML kódu [zdroj vlastní] 37
Obrázek 23: Dialog pro přidání kategorie [zdroj vlastní] 37
Obrázek 24: Příklad tvorby grafu pomocí LiveCharts [zdroj vlastní] 38
Obrázek 25: Ukázka LINQ dotazu [zdroj vlastní] 39
Obrázek 26: Ukázka parametrizovaného dotazu [zdroj vlastní] 40
Obrázek 27: Relační model databáze [zdroj vlastní] 41

Seznam zkratek a značek

API	Application Programming Interface
WPF	Windows Presentation Foundation
LINQ	Language Integrated Query
HTTP	Hyper Text Transfer Protokol
SQL	Structured Query Language
URL	Uniform Resource Locator
JSON	JavaScript Object Notation
XML	Extensible Markup Language
XAML	Extensible Application Markup Language
SRP	Single Responsibility Principle
UI	User Interface
CET	Central European Time
UTC	Coordinated Universal Time

Terminologie

UI

Uživatelské rozhraní je sada vizuálních prvků, které umožňují uživatelům interagovat s aplikací. Dobře navržené UI je zásadní pro použitelnost a uživatelskou spokojenost, jelikož zvyšuje intuitivnost a efektivitu aplikace. Dostupnost UI zajišťuje, že aplikace je přístupná všem uživatelům, včetně těch s omezenými schopnostmi.

Framework

Strukturované programovací prostředí, které podporuje vývoj aplikací, poskytuje knihovny, nástroje a API pro běžné úkoly, a umožňuje vývojářům se soustředit na specifické funkce aplikace místo na základní infrastrukturu.

SHA256

SHA-256 je kryptografická hashovací funkce patřící do rodiny SHA-2 algoritmů. Tento algoritmus převádí vstupní data libovolné délky na pevně daný 256bitový (32 bajtů) hash, což je unikátní digitální otisk těchto dat. SHA-256 je známý svou odolností vůči kolizím, což znamená, že je extrémně náročné najít dvě různé vstupy, které by produkovaly stejný výstup.

SOLID

Sada návrhových principů SOLID, formulovaná Robertem C. Martinem, je akronymem pěti klíčových pravidel pro objektově orientované programování. První princip, Princip jedné odpovědnosti (**S**), uvádí, že třídy by měly mít pouze jednu zodpovědnost nebo jeden důvod pro změnu. Druhý, Princip otevřenosti a uzavřenosti (**O**), doporučuje, aby třídy byly otevřené k rozšíření, ale uzavřené k modifikacím. Liskovův princip zaměnitelnosti (**L**) stanoví, že podtřídy by měly být plně zaměnitelné za své bázové třídy. Princip oddělení rozhraní (**I**) preferuje více specifických rozhraní před jedním univerzálním. Princip obrácení závislostí (**D**) zdůrazňuje, že závislosti by měly existovat na abstrakcích, nikoli na konkrétních implementacích, což znamená, že konkrétnější třídy by měly záviset na abstraktnějších, nikoli naopak.

Úvod

Tato bakalářská práce se zaměřuje na vývoj a implementaci desktopové aplikace pro sledování osobních výdajů a investic. V dnešní době, kdy ekonomická situace vyžaduje pečlivé hospodaření s osobními financemi, se stává správa výdajů nezbytným nástrojem pro dosažení finanční stability a růstu.

Cílem této práce je tvorba uživatelsky přívětivé aplikace, která bude uživatelům umožňovat efektivně zaznamenávat a analyzovat jejich osobní výdaje, poskytovat přehledné grafy, které pomohou v lepším rozhodování o finančních záležitostech, dále sledovat aktuální stav investičních kryptoměnových trhů a evidenci svých nákupů v této oblasti, které se budou hodnotit v závislosti na aktuálním stavu trhu.

Aplikace je zároveň navržena tak, aby byla intuitivní a rychlá v ovládání. Uživatelé budou moci přizpůsobit aplikaci svým specifickým potřebám, například tvorbou vlastních kategorii výdajů nebo i změnu typů zobrazovaných grafů, což umožní lepší orientaci v jejich finanční situaci.

Důležitým aspektem této aplikace je její schopnost poskytovat aktuální informace kryptoměnového trhu pomocí integrace API, která umožňují čerpat nejnovější data o cenách, tržních kapitalizacích a ostatních relevantních finančních ukazatelích v reálném čase a přenášet je do grafických komponent této aplikace.

Aplikace je napsaná v programovacím jazyce C#, kvůli jeho popularitě na trhu a obzvláště kvůli jeho jednoduchosti práce s grafickými komponentami potřebnými pro tvorbu UI. Cílová skupina této aplikace není pevně stanovena, může jí využívat každý, kdo se jakýmkoliv způsobem zajímá o přehled svých financí či kryptoměnové investice.

1 Analýza požadavků

1.1 Funkční požadavky

Správa účtu

Aplikace umožní uživateli vytvořit si uživatelský profil, ke kterému se bude moci následně přihlašovat. S účtem se budou pojit data z uživatelských financí a kryptoměn. Bude ukládán do přenosné databáze, která se bude nacházet v souborech aplikace.

Zobrazení profilových informací

Aplikace bude sledovat uživatelské informace a vypisovat je do sekce profil. Informace se budou skládat z uživatelského jména, přihlašovacího jména, datum posledního přihlášení pro sledování aktivity daného profilu a v neposlední řadě informace o datumu poslední uživatelské evidence dat a počtu vlastněných kryptoměn.

Správa osobních financí

Aplikace umožní přidávat nové výdaje a příjmy, editovat a mazat stávající záznamy. Uživatelé budou moci přiřazovat kategorie k jednotlivým transakcím pro lepší organizaci a sledování.

Správa kryptoměn

V sekci kryptoměny bude možné přidávat a mazat informace o evidovaných kryptoměnách. U každé kryptoměny bude evidován její název, datum nákupu, počet nakoupených jednotek a nákupní cena. Vzhledem k limitovanosti získáváni dat z poskytnutého Coincap API, tento proces vkládání kryptoměn nebude nijak ošetřen validací ceny dané kryptoměny vůči datumu a počtu jednotek.

Převodník

Aplikace bude obsahovat převodník, který bude umožňovat převod měn získávaných z poskytnutého Frankfurter API, které poskytuje kurzy získávané přímo z Centrální Evropské Banky obnovované každý den.

Generování finančních přehledů

Aplikace bude využívat algoritmy ke generování měsíčních, kvartálních a ročních přehledů financí data ukládaná do přenosné databáze, které budou brát v úvahu výdaje a kategorie. Tyto přehledy budou reprezentovány jako grafy, které pomohou uživatelům lépe vizualizovat jejich finanční situaci.

1.2 Nefunkční požadavky

Intuitivní šetření uživatelských chyb

Aplikace by měla být schopna uživatele upozornit, pokud někde zadává nevalidní hodnoty a nemělo by to ovlivnit její chod. Upozornění na takovéto chyby by mělo být ve formě stručných chybových hlášek a dialogů, aby uživatel pochopil, kde udělal chybu.

Interaktivní a jednoduché UI

UI by mělo být navrženo tak, aby bylo co nejjednodušší na pochopení i pro ty, kdo s podobným softwarem nemají předchozí zkušenosti. To zahrnuje logické uspořádání prvků, jasné a konzistentní označení funkcí a minimalizaci počtu kroků potřebných k provedení běžných úkolů. Rozhraní by mělo být také vizuálně přitažlivé, což může zvýšit celkovou spokojenost uživatelů.

Možnosti rozšíření

V rámci tvorby a návrhu aplikace je nutno zohlednit a evidovat možnosti budoucího rozšíření.

Škálovatelnost

Aplikace musí být navržena tak, aby byla snadno rozšiřitelná v budoucnosti bez nutnosti kompletního přepisu existujícího kódu. To zahrnuje modularitu kódu, využití snadno rozšiřitelných technologií a datových struktur, a kapacitu pro integraci nových modulů nebo rozšíření stávajících funkcí.

Výkon

Aplikace by měla být optimalizována pro rychlou odezvu s důrazem na asynchronní zpracování dat. Veškeré načítání dat z externích API by mělo probíhat asynchronně, aby nedošlo k blokování uživatelského rozhraní. To znamená, že uživatel by měl mít možnost pokračovat v používání aplikace i během načítání dat. Asynchronní operace by měly být správně spravovány a integrovány do GUI, tak aby aktualizace dat v rozhraní byly plynulé a bez viditelného zpoždění. Tento přístup zlepšuje celkový uživatelský zážitek tím, že minimalizuje čekací doby a zvyšuje reaktivitu aplikace.

2 Teoretický úvod

2.1 Kryptoměna

Kryptoměna představuje digitální formu měny, kterou lze označit za elektronické peníze reprezentující moderní typ měnového systému. Podle vyjádření Generálního finančního ředitelství a rozhodnutí krajského soudu v Brně však kryptoměna není považována za tradiční měnu nebo peníze, ale spíše za nehmotnou movitou věc. Tyto digitální „peníze“ jsou vytvářeny elektronicky a základní technologií, která umožňuje jejich fungování, je asymetrická kryptografie. Nejstarším, nejznámějším a zároveň nejdražším představitelem kryptoměny je Bitcoin. Kryptoměny bývají taktéž označeny a reprezentovány pojmem tokeny. [1]

2.1.1 Bitcoin

Bitcoin (BTC) je forma digitální měny, známá jako kryptoměna, která nevyžaduje centrální banku ani správce. Umožňuje přímý přenos mezi uživateli s využitím šifrovacích technologií a je zaznamenáván v technologii zvané blockchain. Bitcoin byl vytvořen v roce 2009 a je dostupný jako open-source software. Jeho vývojáři jsou známí pod pseudonymem Satoshi Nakamoto, Tato kryptoměna se používá pro obchody s jinými měnami a pro nákupy zboží a služeb, přičemž nabízí vysokou míru anonymity a těžko sledovatelných transakcí. Bitcoiny lze získat koupí nebo těžbou, což je proces generování nových bloků v blockchainu. Těžaři za tuto aktivitu obdrží odměnu v podobě částí tokenů Bitcoinu. Těžba je výpočetně náročná, a kromě silného hardwaru, obvykle grafických karet vhodných pro tyto účely, vyžaduje také značné množství energie. [2]

2.2 Asymetrická kryptografie

Asymetrická kryptografie, známá také jako kryptografie s veřejným klíčem, se začala vyvíjet od 70. let 20. století a představuje v oblasti informačních technologií typ algoritmu, který pracuje s páry klíčů: veřejným a soukromým. Veřejný klíč je možné ověřit skrze digitální certifikát a je určen k volné distribuci, což umožňuje jeho využití k šifrování zpráv nebo k ověřování digitálních podpisů. Soukromý klíč, který by měl zůstat tajný a dostupný pouze držiteli klíče, se používá pro dešifrování zpráv nebo pro tvorbu digitálních podpisů. [3]

2.3 Token

Token je digitální reprezentace různých forem majetku nebo práv, uložená na blockchainové síti, což zvyšuje jeho bezpečnost, snižuje možnost falšování a usnadňuje jejich širší distribuci a uznání. Jeho specifická role a význam jsou určeny emitentem, což může být hodnota, majetkový podíl, hlasovací právo, nebo zástupce hmotné či nehmotné věci. Tokeny tak představují digitální aktiva a symboly vlastnictví. Tokeny lze také využít k zjednodušení a automatizaci transakcí, protože fungují na decentralizovaných platformách, které umožňují rychlé a transparentní převody bez potřeby tradičních finančních zprostředkovatelů. Dále slouží jako základní stavební bloky pro vývoj decentralizovaných aplikací a inteligentních kontraktů, což rozšiřuje jejich využití napříč různými odvětvími a obchodními modely. [4]

2.4 Blockchain

Blockchain je typ distribuované a decentralizované databáze, která je přístupná jakémukoli uživateli s internetovým připojením. Termíny „decentralizovaná“ a „distribuovaná“ zde vyjadřují, že tato technologie nemá ústřední kontrolní bod, není ovládána žádnou centrální autoritou a nezávisí na žádném jednotlivém rozhodnutí provozovatele. To znamená, že v síti blockchainu mají všichni uživatelé stejná práva a možnosti. Veškeré změny dat v databázi, ať už jde o jejich odstranění nebo přidání nových, musí být schváleny všemi uzly v síti. Tento proces zajišťuje, že všechny transakce zaznamenané na blockchainu jsou bezpečné a důvěryhodné, neboť není možné manipulovat s daty ve prospěch jednotlivce. Transakce jsou na blockchainu organizovány do bloků, což vysvětluje první část názvu technologie. Tyto bloky jsou následně ověřovány síťovými uzly, tedy ostatními uživateli, a pokud jsou shledány správnými, jsou připojeny k předchozím blokům v databázi. Proces propojování bloků vytváří neustále rostoucí řetězec (chain) bloků, který je zcela transparentní a nemožný zfalšovat. Úspěšné připojení nového bloku dat k existujícímu řetězci je často odměňováno v podobě tokenů. [5]

2.5 Tržní pojmy

Objem

Objem obchodování obvykle reflektuje celkovou hodnotu nebo množství daných aktiv, které byly během daného časového období obchodovány na burze nebo v rámci termínového trhu. Objem obchodů, zaznamenaný s přesnými referenčními čísly, ukazuje míru aktivity investorů na trhu v daném okamžiku, navíc, objem obchodů může poskytnout důležité informace o likviditě trhu, kde vyšší objem obvykle značí vyšší likviditu, což umožňuje investorům lépe vstupovat a vystupovat z pozic bez významného ovlivnění cen aktiv. [6]

Tržní kapitalizace

Tržní kapitalizace představuje současnou tržní hodnotu společnosti a vypočítává se jako produkt celkového počtu aktiv, které společnost vydala, a aktuální ceny těchto aktiv na trhu. Tento ukazatel je klíčovým faktorem pro zařazení společnosti do akciových indexů, jako je například S&P 500 ve Spojených státech, neboť odráží ekonomickou velikost a význam dané společnosti na trhu. Změny v tržní kapitalizaci mohou také naznačovat, jak trh vnímá budoucí růst nebo rizika spojená se společností, což je klíčové pro investiční rozhodování a strategické plánování. [7]

Maximální nabídka

Maximální nabídka se vztahuje na nejvyšší možný počet jednotek, které mohou být kdy vyrobeny nebo vydány v rámci dané kryptoměny. Tento limit je často stanoven při tvorbě měny a je zapsán do jejího protokolu. Například, Bitcoin má maximální nabídku 21 milionů mincí, což znamená, že více než 21 milionů Bitcoinů nemůže být nikdy vytvořeno. Tento koncept je zásadní pro pochopení deflačních vlastností některých kryptoměn, které mohou teoreticky zvyšovat jejich hodnotu v čase s omezenou nabídkou. Omezení maximální nabídky, jako v případě Bitcoinu, často vytváří percepci vzácnosti, která může vést k zvyšování ceny, pokud poptávka převýší dostupnou nabídku, což je běžný jev v tržních cyklech kryptoměn. [8]

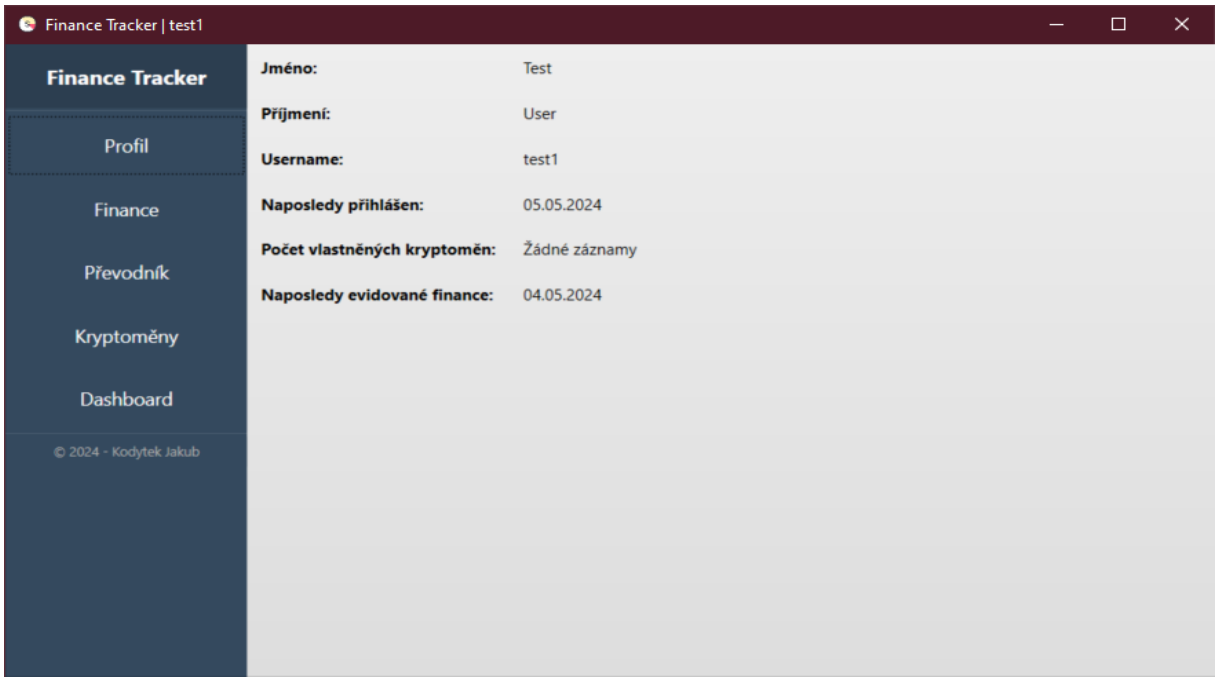
3 Struktura aplikace

Aplikace se dá používat pouze po úspěšné registraci či přihlášení. Při registraci či přihlašování je použito šifrování hesla pomocí funkce SHA256, z databáze tedy není možno získat heslo v nezašifrované formě. Toto je minimální, ale efektivní zabezpečení, které tato aplikace používá. Pokud se uživatel zaregistruje, je mu pomocí třídy DatabaseContentService vytvořena databázová struktura, konkrétně je zaevidován do tabulky Users se svými přihlašovacími a osobními údaji, vytvoří se mu základní kategorie financí v tabulce UserCategories, které může používat, upravovat či mazat a založí se mu prostor ve formě tabulek pro evidenci jeho osobních financí (UserFinances) a kryptoměnových investic (UserCryptos).

3.1 Moduly aplikace

3.1.1 Profil

Modul profil v této aplikaci zobrazuje uživatelovi jeho osobní data ve formě textu, k těmto datům je přidán datum jeho posledního přihlášení do aplikace a datum poslední evidence finančních záznamů. Společně s osobními a informativními daty se zde zobrazuje počet jeho vlastněných kryptoměn, který se týká čistě vlastněných kryptoměn, nikoliv evidovaných transakčních záznamů dané kryptoměny, tj. například kdyby uživatel evidoval 3 transakce s kryptoměnou Bitcoin, vlastněné kryptoměny na profilu budou mít hodnotu 1.

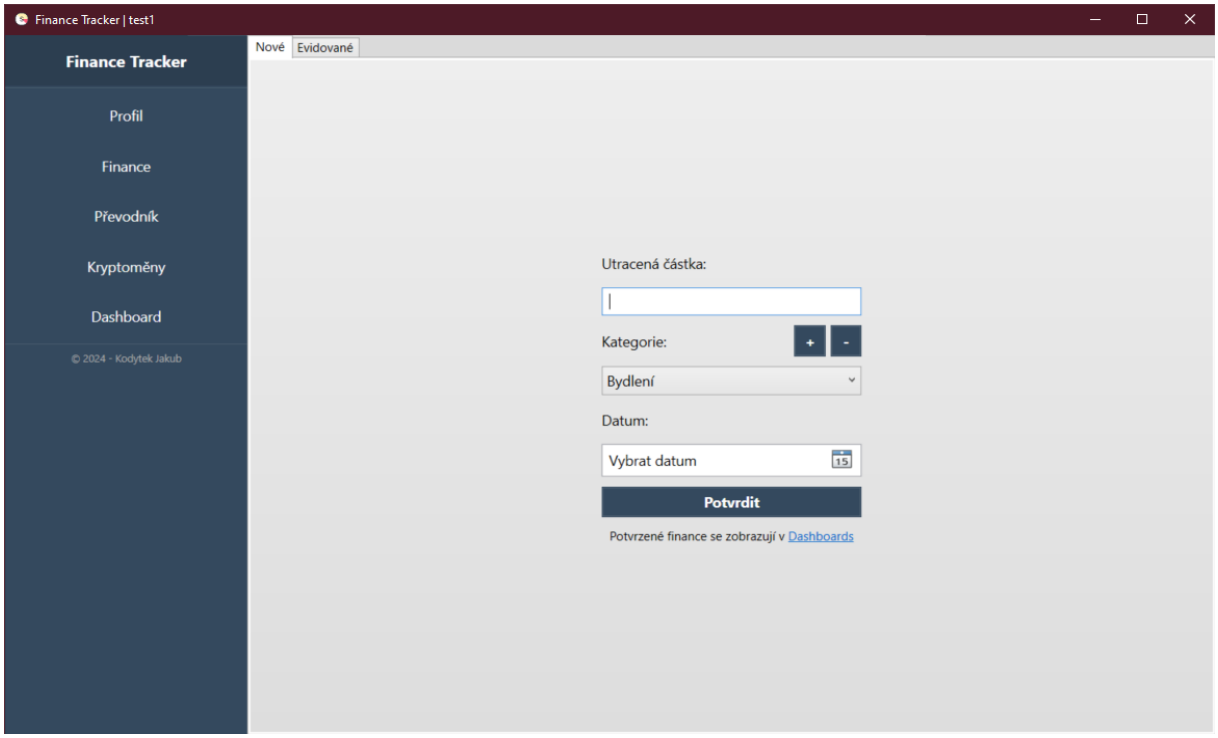


Obrázek 1: Modul profil [zdroj vlastní]

3.1.2 Finance

3.1.2.1 Záložka nové

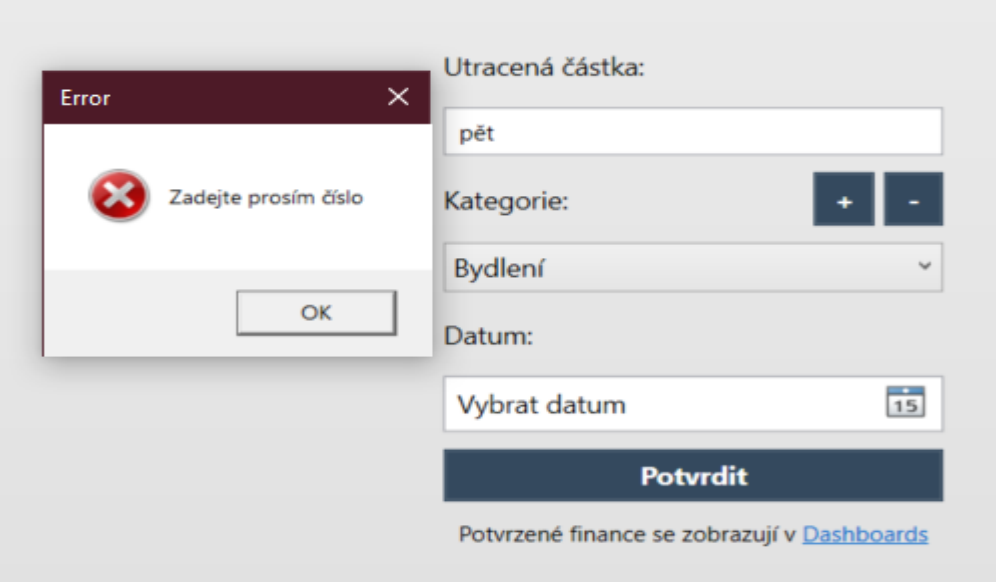
Záložka modulu finance umožňuje uživateli vytvářet nové finanční záznamy. Finanční záznam se skládá z utracené částky, kategorie a datumu. Po vložení validních dat se záznam uloží do databáze UserFinances, ze které je poté zpracováván v modulu Dashboard.



Obrázek 2: Modul Finance – záložka nové [zdroj vlastní]

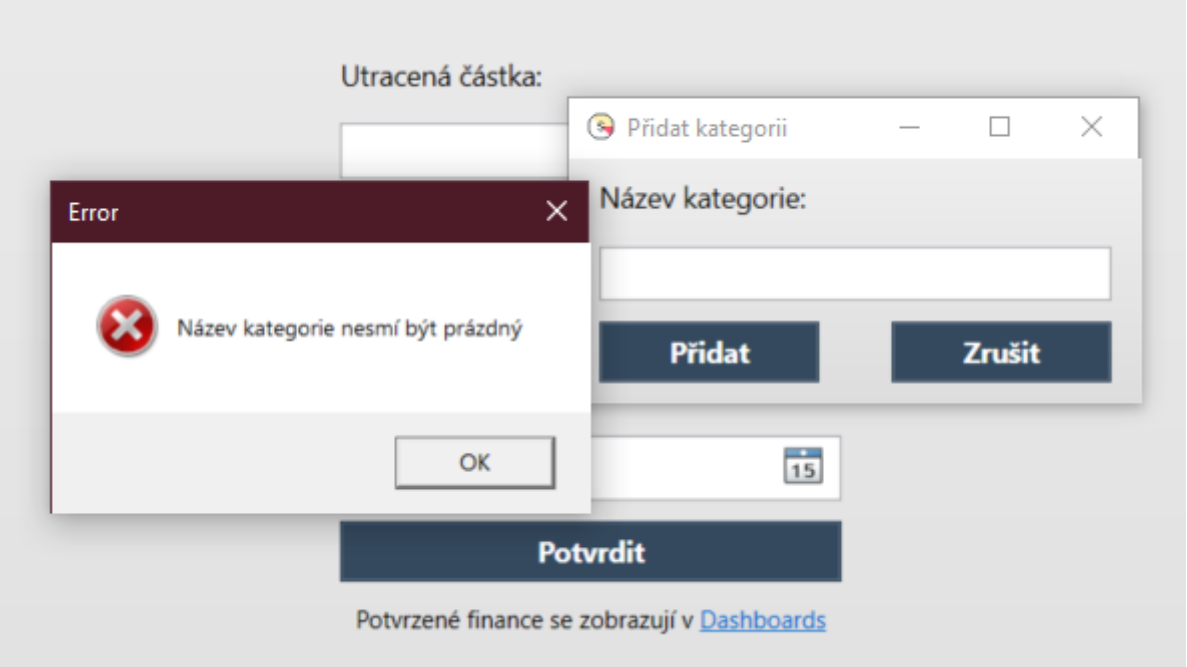
Jsou zde přítomny validace uživatelského vstupu

- Pole utracená částka, která zajišťuje, že uživatel zadá do pole pouze číselnou hodnotu



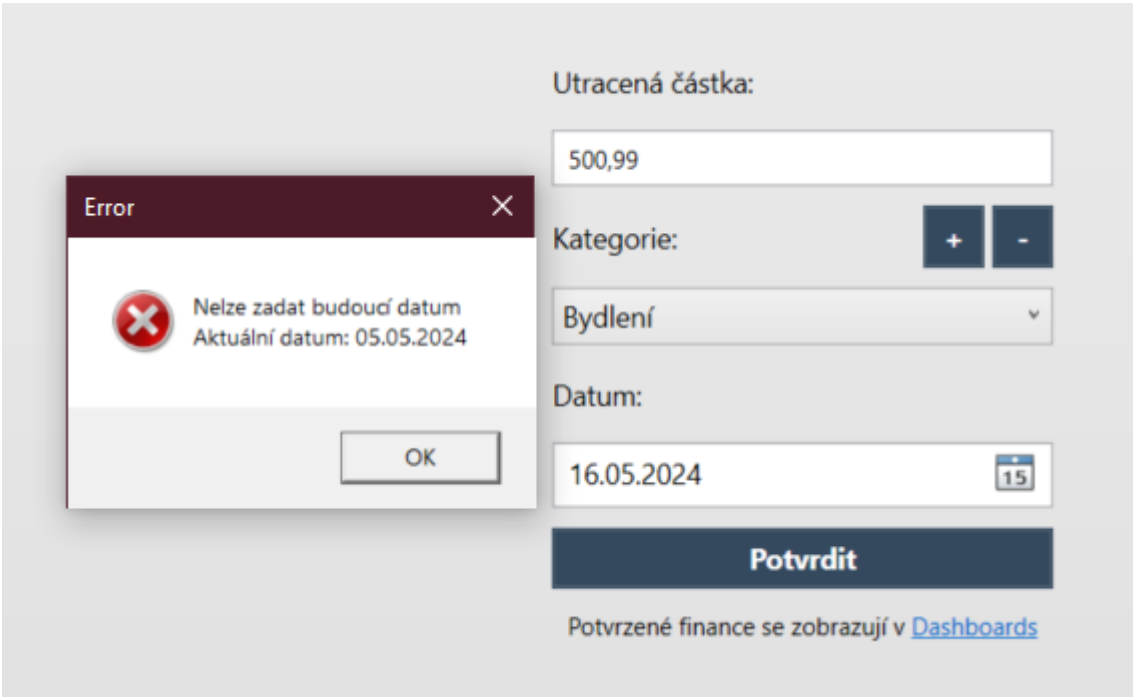
Obrázek 3: Modul Finance – validace částky [zdroj vlastní]

- Při vytváření nové kategorie nesmí obsahovat prázdný řetězec



Obrázek 4: Modul Finance – validace názvu kategorie [zdroj vlastní]

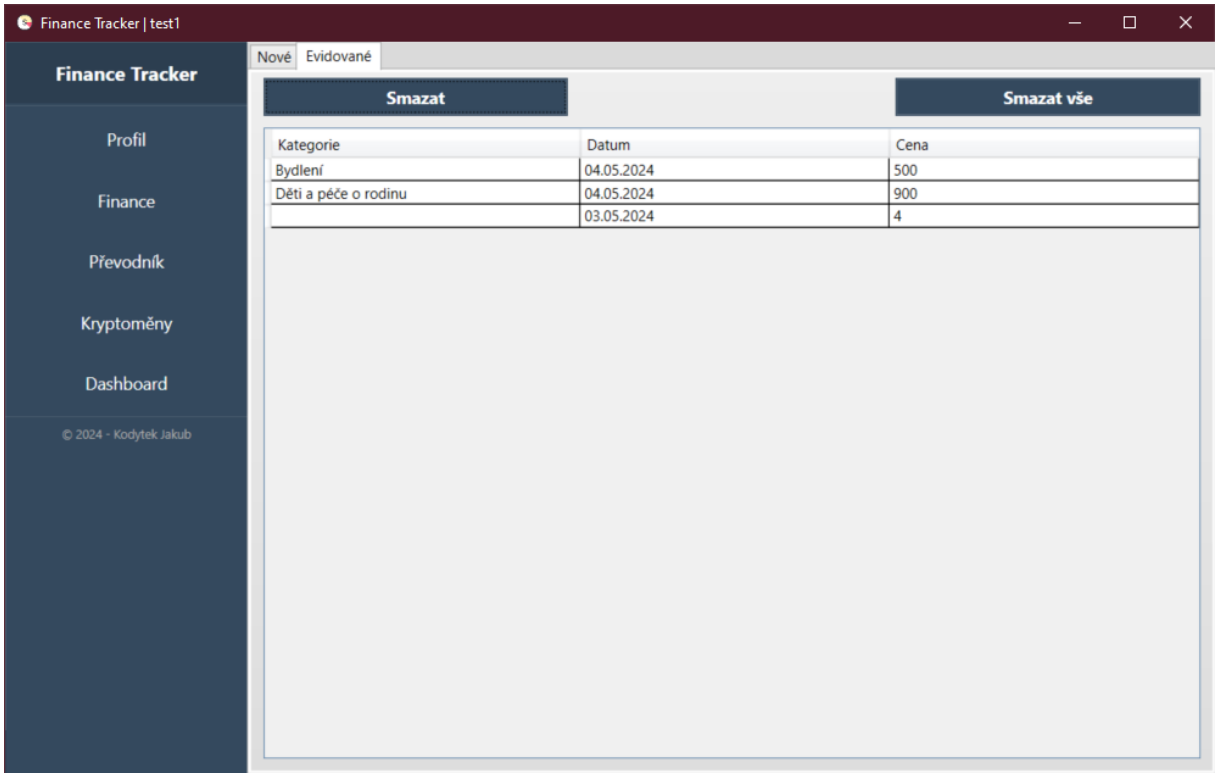
- Při zadávání datumu evidence není možné vybrat budoucí datum



Obrázek 5: Modul Finance – validace datumu [zdroj vlastní]

3.1.2.2 Záložka evidované

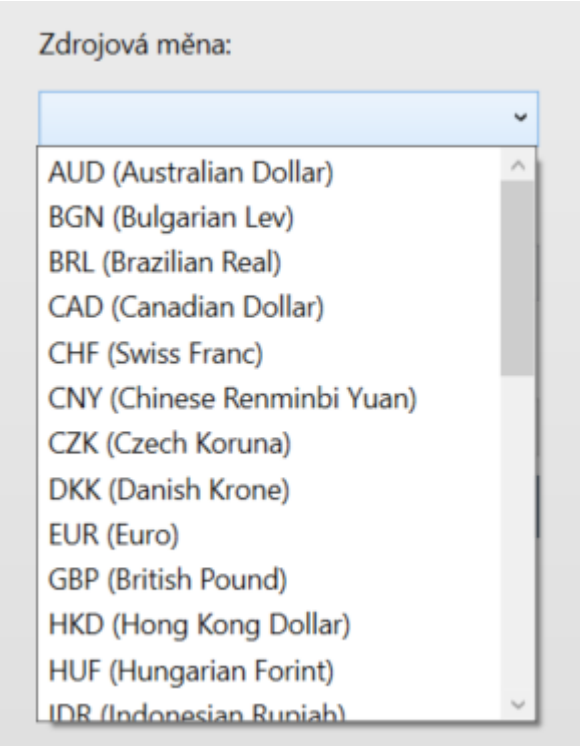
Záložka evidované v modulu finance je klíčovým nástrojem pro uživatele, který umožňuje efektivně spravovat a přehledně zobrazovat všechny finanční transakce zaznamenané v aplikaci. Uživatelé mohou procházet jednotlivé záznamy, které jsou rozděleny podle kategorií, data a ceny, což usnadňuje rychlý přehled o jejich finančním toku. Kromě toho záložka nabízí možnost „hromadného mazání", kde uživatel může efektivně odstranit záznamy, které již nejsou potřebné. Tato funkce je zvlášť užitečná pro udržování čistoty a aktuálnosti databáze, protože umožňuje odstranit staré nebo nepotřebné finanční záznamy podle přesně definovaných kritérií. Tlačítko „smazat“ spouští selektivní dotaz do databáze, který zaručuje, že bude smazán pouze záznam, který odpovídá všem viditelným atributům. To značně zvyšuje efektivitu práce s daty a minimalizuje možnost chybného smazání důležitých informací.



Obrázek 6: Modul finance – záložka evidované [zdroj vlastní]

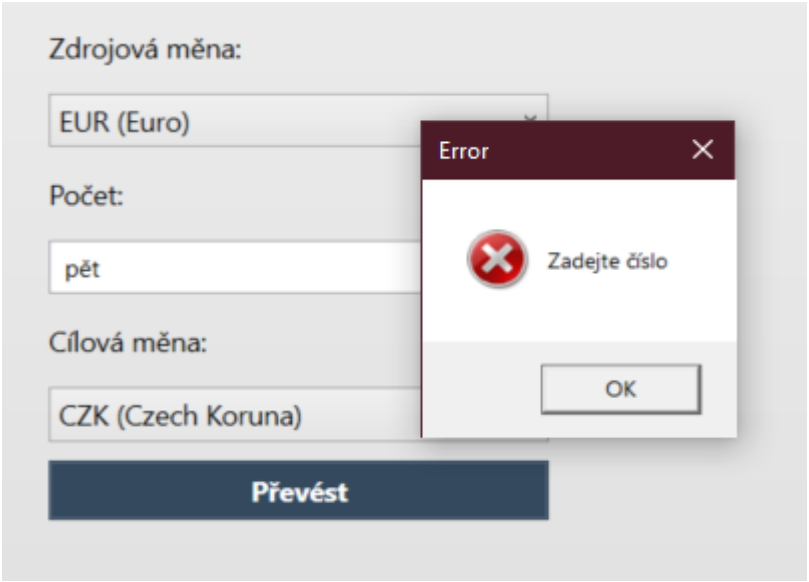
3.1.3 Převodník

Modul převodník slouží k převodu dvou měn, obsahuje 2 combo boxy se zdrojovou a cílovou měnou a pole pro hodnotu měny, která se převádí. Combo boxy jsou asynchronně plněny daty asynchronně získanými z Frankfurter API, které poskytuje informace o dostupných měnách pro převody a jejich aktuálních převodních kurzů.



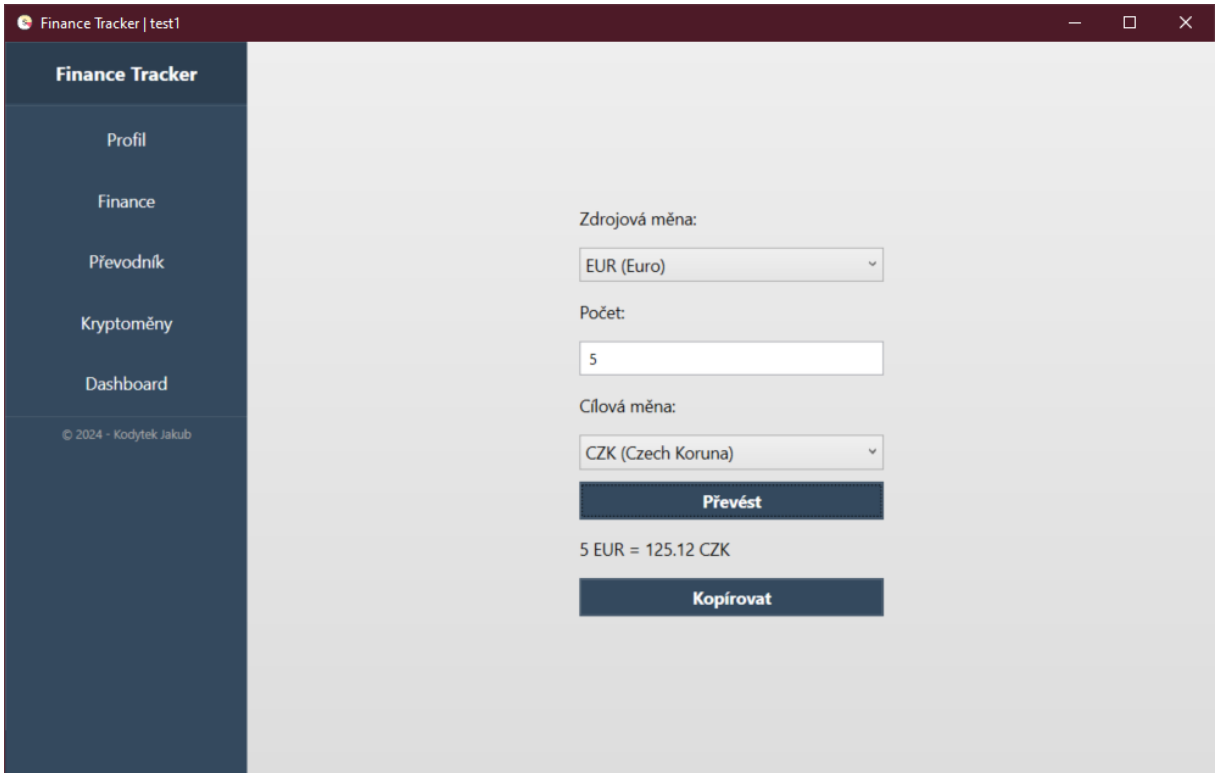
Obrázek 7: Modul Převodník – ukázka naplněného combo boxu [zdroj vlastní]

Je zde přítomna validace uživatelského vstupu do pole hodnoty pro převod.



Obrázek 8: Modul Převodník – validace hodnoty [zdroj vlastní]

Po úspěšném zadání celo(číselné) hodnoty do pole pro hodnotu převodu je zobrazen výsledný popisek s aktuálním kurzem převodu a tlačítko kopírovat, které umožňuje tento výsledek zkopírovat do uživatelské schránky.



Obrázek 9: Modul Převodník – ukázka úspěšného převodu [zdroj vlastní]

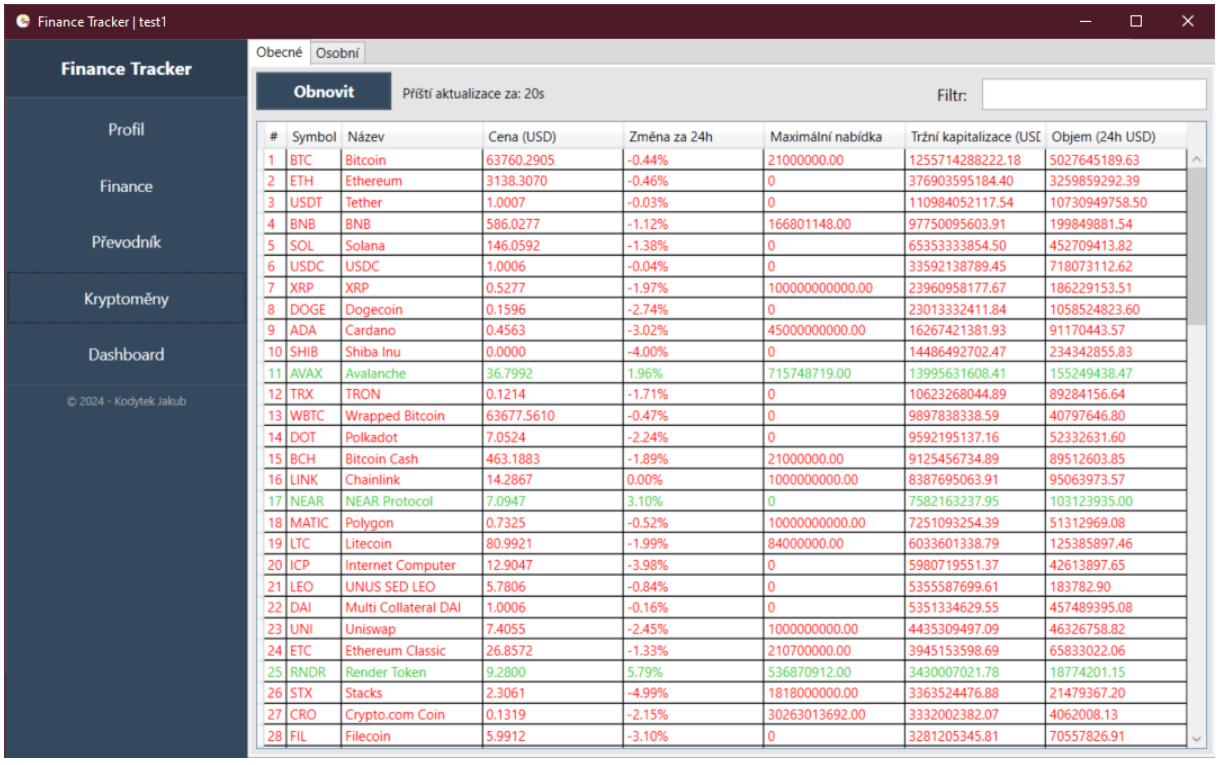
3.1.3.1 Frankfurter API

Toto API slouží k získávání informací o dostupných měnách a jejich aktuálních převodních kurzech. API získává data o referenčních směnných kurzech zveřejněné přímo z Evropské centrální banky, díky čemuž poskytuje věrohodné a přesné údaje svým uživatelům. Data jsou aktualizována každý den přibližně v 16:00 CET (středoevropský čas), což je o hodinu napřed oproti UTC (koordinovaný světový čas). [9]

3.1.4 Kryptoměny

3.1.4.1 Záložka obecné

Modul kryptoměny, záložka obecné, slouží k zobrazení aktuálně dostupných informací z CoinCap API. Zobrazují se zde informace o pořadí, symbolu a názvu kryptoměny, její aktuální tržní ceny v USD, procentuální denní změně v ceně, maximální nabídce dané kryptoměny, tržní kapitalizace a prodaném objemu v měně USD za poslední den. Data jsou asynchronně získávána z API, přeformátována pomocí knihovny Json.NET a následně asynchronně přidávána do Data Gridu, dále jsou obarvena dle hodnoty procentuální změny. Červená barva značí negativní procentuální změnu a zelená kladnou procentuální změnu. Je zde i pole filtr, které umožňuje uživateli vyhledat danou kryptoměnu podle pořadí, symbolu či názvu kryptoměny. Jako poslední se zde nachází tlačítko obnovit, které znovu načte dostupné informace z API a obnoví i popisek, který označuje, kdy dojde k dalšímu obnovení, ten je standardně nastaven na 20s, ale je možno ho v konfiguračním souboru aplikace změnit.

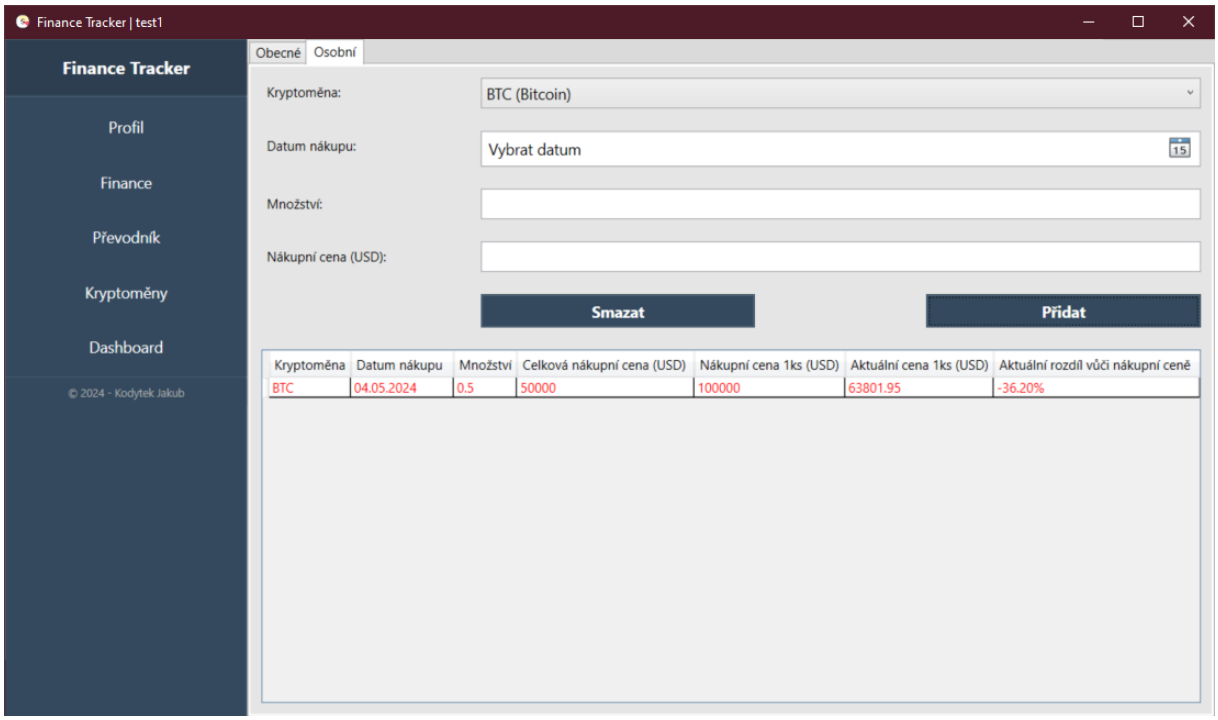


#	Symbol	Název	Cena (USD)	Změna za 24h	Maximální nabídka	Tržní kapitalizace (USD	Objem (24h USD)
1	BTC	Bitcoin	63760.2905	-0.44%	21000000.00	1255714288222.18	5027645189.63
2	ETH	Ethereum	3138.3070	-0.46%	0	376903595184.40	3259859292.39
3	USDT	Tether	1.0007	-0.03%	0	110984052117.54	10730949758.50
4	BNB	BNB	586.0277	-1.12%	166801148.00	97750095603.91	199849881.54
5	SOL	Solana	146.0592	-1.38%	0	65353333854.50	452709413.82
6	USDC	USDC	1.0006	-0.04%	0	33592138789.45	718073112.62
7	XRP	XRP	0.5277	-1.97%	100000000000.00	23960958177.67	186229153.51
8	DOGE	Dogecoin	0.1596	-2.74%	0	23013332411.84	1058524823.60
9	ADA	Cardano	0.4563	-3.02%	45000000000.00	16267421381.93	91170443.57
10	SHIB	Shiba Inu	0.0000	-4.00%	0	14486492702.47	234342855.83
11	AVAX	Avalanche	36.7992	1.96%	715748719.00	13995631608.41	155249438.47
12	TRX	TRON	0.1214	-1.71%	0	10623268044.89	89284156.64
13	WBTC	Wrapped Bitcoin	63677.5610	-0.47%	0	9897838338.59	40797646.80
14	DOT	Polkadot	7.0524	-2.24%	0	9592195137.16	52332631.60
15	BCH	Bitcoin Cash	463.1883	-1.89%	21000000.00	9125456734.89	89512603.85
16	LINK	Chainlink	14.2867	0.00%	1000000000.00	8387695063.91	95063973.57
17	NEAR	NEAR Protocol	7.0947	3.10%	0	7582163237.95	103123935.00
18	MATIC	Polygon	0.7325	-0.52%	10000000000.00	7251093254.39	51312969.08
19	LTC	Litecoin	80.9921	-1.99%	84000000.00	6033601338.79	125385897.46
20	ICP	Internet Computer	12.9047	-3.98%	0	5980719551.37	42613897.65
21	LEO	UNUS SED LEO	5.7806	-0.84%	0	5355587699.61	183782.90
22	DAI	Multi Collateral DAI	1.0006	-0.16%	0	5351334629.55	457489395.08
23	UNI	Uniswap	7.4055	-2.45%	1000000000.00	4435309497.09	46326758.82
24	ETC	Ethereum Classic	26.8572	-1.33%	210700000.00	3945153598.69	65833022.06
25	RNDR	Render Token	9.2800	5.79%	536870912.00	3430007021.78	18774201.15
26	STX	Stacks	2.3061	-4.99%	1818000000.00	3363524476.88	21479367.20
27	CRO	Crypto.com Coin	0.1319	-2.15%	30263013692.00	3332002382.07	4062008.13
28	FIL	Filecoin	5.9912	-3.10%	0	3281205345.81	70557826.91

Obrázek 10: Modul Kryptoměny – záložka obecné [zdroj vlastní]

3.1.4.2 Záložka osobní

Modul kryptoměny, záložka osobní, umožňuje uživateli evidovat jeho osobní kryptoměnové transakce. Nachází se zde combo box s asynchronně načtenými kryptoměnami z CoinCap API, pole datum nákupu, pole množství, které specifikuje část nakoupeného tokenu a pole nákupní cena v USD. Při úspěšném evidování uživatelské transakce je záznam přidán do databáze a zároveň do Data Gridu, ve kterém se obarví dle aktuálního procentuálního rozdílu ceny kryptoměny vůči nákupní ceně kryptoměny. Z dat získaných od uživatele se použije nákupní cena vůči danému množství tokenu a z ní se vypočítá nákupní cena jednoho tokenu, z které se poté počítá daný procentuální rozdíl vůči nákupní ceně.



Obrázek 11: Modul Kryptoměny – záložka osobní [zdroj vlastní]

Existují zde validace uživatelských hodnot, kontroluje se správnost datumu, tedy zda uživatel nezadal budoucí datum, validní (celo)číselná hodnota v poli množství a validní hodnota v poli nákupní cena, na všechny nevalidní hodnoty je uživatel upozorněn chybovými dialogy, stejně jako například v modulu Převodník.

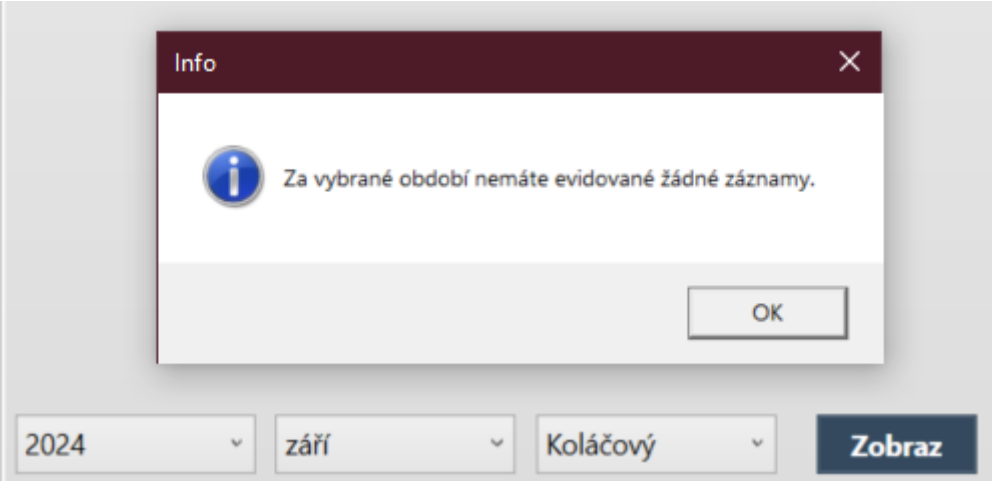
3.1.4.3 CoinCap API

Toto API slouží k získávání informací o dostupných kryptoměnách a jejich aktuálních informacích, jako jsou tržní kurzy, prodané množství tokenů za 24h atd. API získává data z tisíců trhů a díky tomu je schopno poskytovat transparentní a přesné údaje o cenách a dodatečných informacích dané kryptoměny. [10]

3.1.5 Dashboard

Modul dashboard slouží k vizualizaci uživatelsky evidovaných financí do grafů. Jsou zde aktuálně dostupné 2 druhy grafů: koláčový a sloupcový. K tvorbě těchto grafů byla použita knihovna LiveCharts, která umožňuje intuitivní přístup k tvorbě vizualizací dat, v mém případě pomocí Data bindings. Při kliknutí na tento modul se automaticky nastaví a zobrazí první záložka – měsíční přehled. Nachází se zde validace dat:

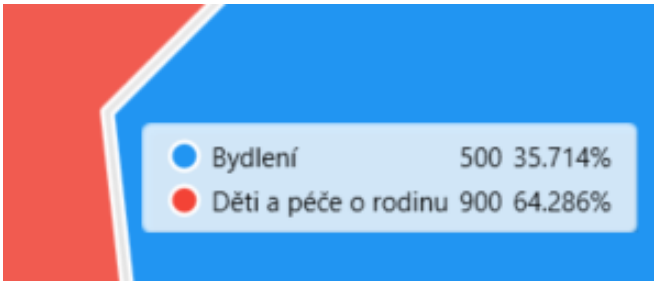
- Pokud uživatel vybere datum, ve kterém nemá evidované žádné záznamy, zobrazí se informační dialog, který ho na tuto skutečnost upozorní.



Obrázek 12: Modul Dashboard – ukázka dialogu při chybném výběru dat [zdroj vlastní]

Tento dialog se zobrazuje napříč celým modulem Dashboard, ať už v záložce měsíční přehled při špatně vybraném roce či měsíci, v záložce kvartální přehled při špatně vybraném roce či kvartálním období a v záložce roční přehled při špatně vybraném roce.

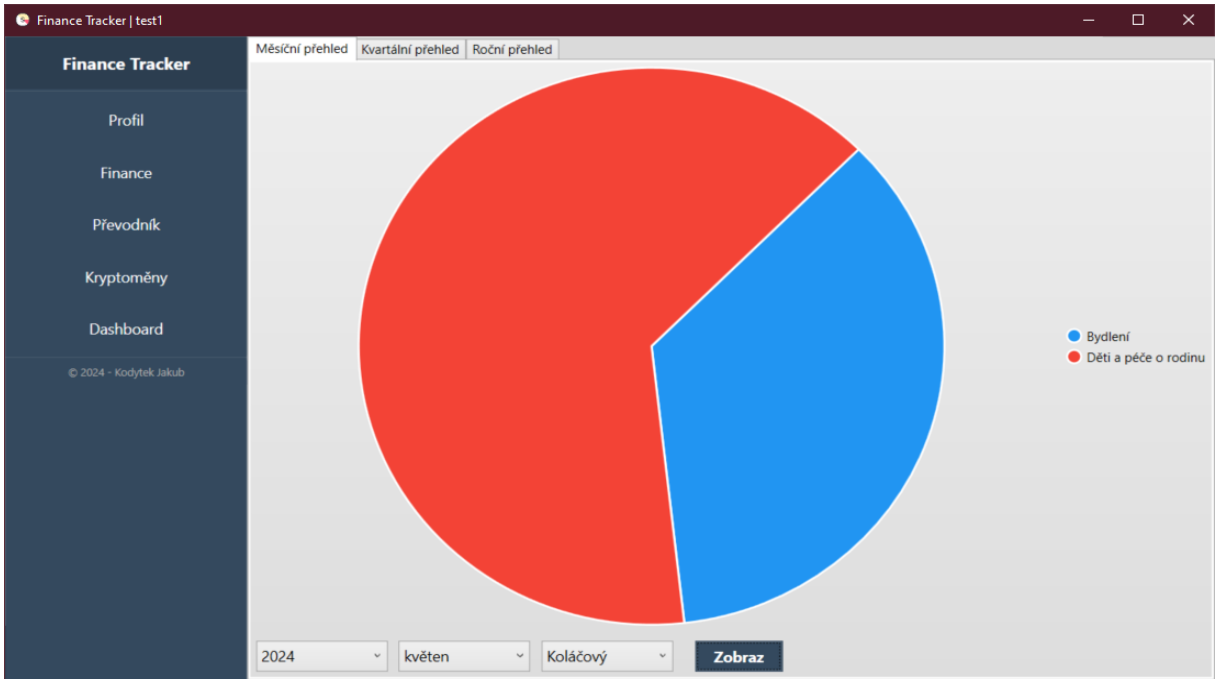
Každý graf obsahuje legendu, ta se zobrazí při najetí kurzorem na libovolnou část grafu. Tato legenda má přednastavený formát z knihovny LiveCharts, dají se ale vytvářet vlastní legendy pro podrobnější či srozumitelnější popis dat.



Obrázek 13: Modul Dashboard – ukázka legendy grafu [zdroj vlastní]

3.1.5.1 Měsíční přehled

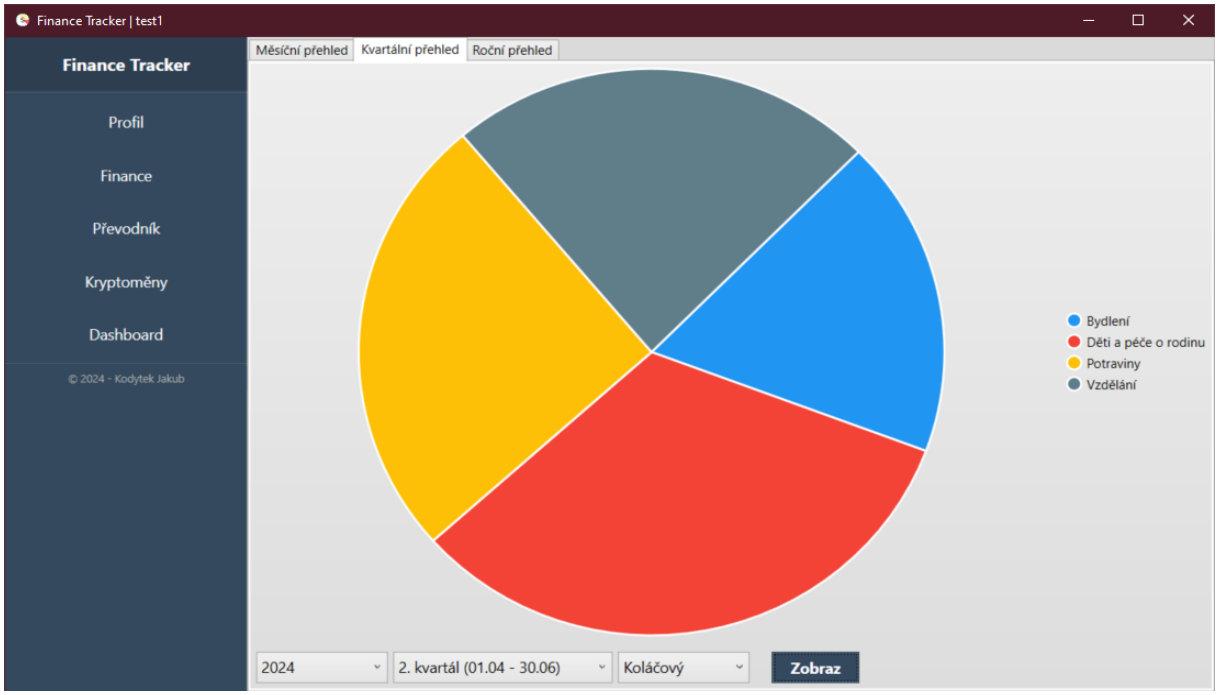
Záložka měsíční přehled obsahuje 3 combo boxy, první obsahuje pole let, které je získáváno z databáze z tabulky UserFinances tak, že se vyberou všechny záznamy pro daného uživatele a vyfiltrují se z datumů evidencí roky. Těmito roky je následně plněn první combo box. Druhý combo box obsahuje měsíce v našem roce, tyto hodnoty jsou získávány z knihovny CultureInfo a následně plněny do druhého combo boxu. Třetí obsahuje typy aktuálně podporovaných druhů grafů, do kterých je aplikace schopna převést uživatelská data a následně je zobrazit.



Obrázek 14: Modul Dashboard – měsíční přehled, koláčový graf [zdroj vlastní]

3.1.5.2 Kvartální přehled

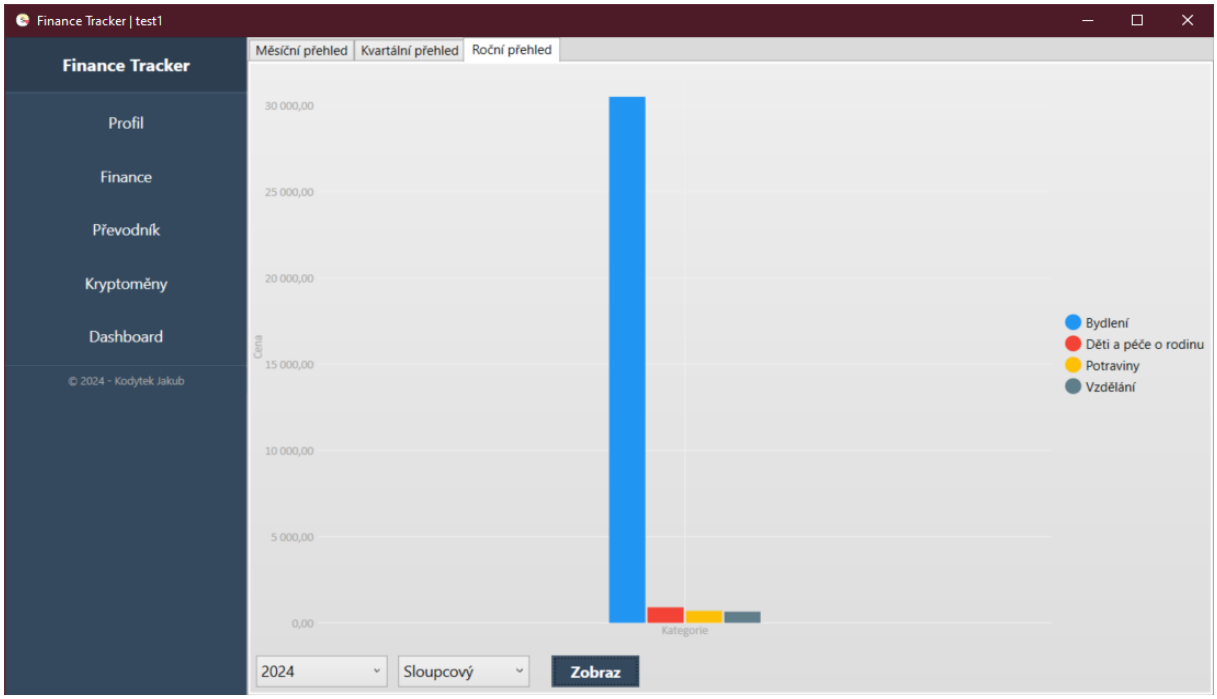
Záložka kvartální přehled poskytuje uživatelům podrobný náhled na jejich finanční tok v průběhu jednotlivých čtvrtletí roku. Obsahuje tři combo boxy: první z nich umožňuje uživatelům vybrat konkrétní rok, druhý nabízí výběr mezi čtyřmi čtvrtletími a třetí combo box umožňuje uživatelům zvolit typ zobrazeného grafu, jako je koláčový nebo sloupcový graf. Toto uspořádání umožňuje efektivní srovnání a analýzu finančních výsledků podle různých časových úseků a grafických reprezentací, což poskytuje uživatelům lepší přehled o vývoji jejich financí a pomáhá identifikovat trendy nebo oblasti pro zlepšení. Grafické zobrazení podporuje rychlé pochopení finančních dat a uživatelské finanční situace.



Obrázek 15: Modul Dashboard – kvartální přehled, koláčový graf [zdroj vlastní]

3.1.5.3 Roční přehled

Záložka roční přehled, integrovaná v modulu Dashboard, poskytuje uživatelům komplexní pohled na jejich finanční aktivitu za celý vybraný rok. Obsahuje dva combo boxy: první umožňuje výběr roku a druhý nabízí možnosti různých typů grafů, jako jsou sloupcové nebo koláčové grafy, což umožňuje uživatelům přizpůsobit zobrazení dat podle svých potřeb. Tato funkce je neocenitelná pro uživatele, kteří potřebují sledovat svůj finanční vývoj a identifikovat klíčové trendy, což jim pomáhá v plánování a efektivním řízení osobních financí. Nabídka různých typů vizualizací umožňuje uživatelům lepší analytické porozumění svým financím, a tím podporuje lepší finanční rozhodování.



Obrázek 16: Modul Dashboard – roční přehled, sloupcový graf [zdroj vlastní]

3.2 Kód

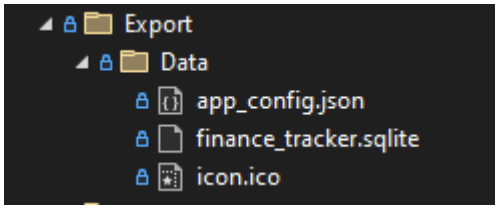
3.2.1 Struktura kódu

V aplikaci nejsou použité žádné návrhové vzory kromě vzoru Singleton. Je strukturovaná pouze do balíčků pro odlišení funkcionality a zpřehlednění struktury kódu. V aplikaci je použit princip SRP, což je jeden ze základních principů objektově orientovaného návrhu známých pod akronymem SOLID.

Popis jednotlivých balíčků:

Export > Data

Tento balíček slouží pro přenos datových souborů aplikace pro případ práce na jiných zařízeních, nachází se zde konfigurační soubor aplikace, databázový soubor a ikona aplikace.

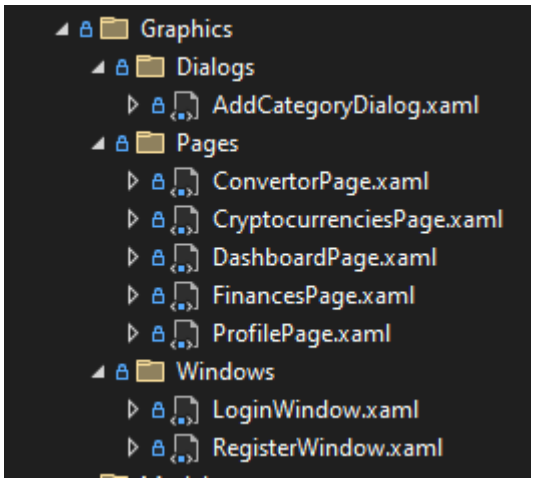


Obrázek 17: Struktura kódu – balíček export [zdroj vlastní]

Graphics

Tento balíček obsahuje grafické soubory aplikace, nachází se zde

- Balíček Dialogs, který obsahuje dialog pro tvorbu nové kategorie v modulu Finance.
- Balíček Pages, který definuje jednotlivé stránky v menu aplikace
- Balíček Windows, který obsahuje okna pro přihlášení a registraci do aplikace

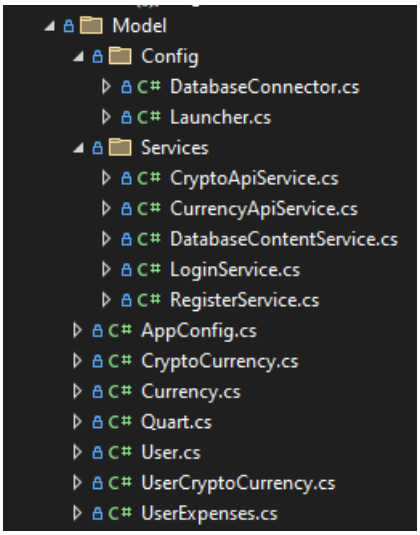


Obrázek 18: Struktura kódu – balíček graphics [zdroj vlastní]

Model

Balíček obsahuje

- Konfigurační třídy v balíčku Config, které pracují s konfiguračním souborem aplikace
- Service třídy, které se starají o zprostředkování hlavních funkcionalit aplikace
- Datové třídy, které slouží pro tvorbu modelu aplikace



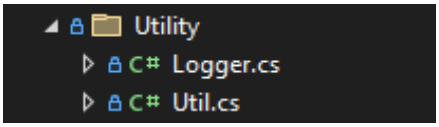
Obrázek 19: Struktura kódu – balíček model [zdroj vlastní]

Stručný popis důležitých tříd:

- **DatabaseConnector** – zajišťuje připojení k databázovému systému SQLite
- **Launcher** – obsahuje sekvenci příkazů nutných pro inicializaci potřebných objektů a spuštění aplikačních částí v definovaném pořadí
- **CryptoApiService** – zajišťuje asynchronní získávání dat z API určeného pro kryptoměny a jejich následnou transformaci
- **CurrencyApiService** – stará se o asynchronní získávání, transformaci a konverzi dat z API určeného pro konverzi měn.
- **DatabaseContentService** – stará se o tvorbu databáze a databázového obsahu ve formě tabulek a relačních vazeb.
- **LoginService** – zajišťuje přihlašování uživatele do databáze.
- **RegisterService** – zajišťuje registraci uživatele do databáze

Utility

Balíček obsahující třídy, které poskytují užitečné funkcionality napříč aplikací



Obrázek 20: Struktura kódu – balíček utility [zdroj vlastní]

3.2.2 Singleton

V aplikaci byl návrhový vzor Singleton použit k zajištění, že instance pro připojení k databázi existuje v aplikaci pouze jednou. Tento vzor je velmi užitečný ve scénářích, kde je potřeba sdílet jedinou instanci objektu napříč celou aplikací, jakým je například připojení k databázi, což zajišťuje efektivní správu zdrojů a eliminuje riziko nekonzistence dat. V jazyce C# je zápis Singletonu možný použitím operátoru „??=“, který zkontroluje, zda je proměná nastavena na hodnotu null. Pokud ano, přiřadí ji nově vytvořenou instanci třídy, pokud ne, vrátí již existující instanci. Tento přístup zajišťuje, že každé další volání této instance bude odkazovat na stejný objekt. Důležité je, že proměnná držící instanci Singletonu by měla být deklarována jako static, což znamená, že je sdílena mezi všemi instancemi třídy, čímž se zajišťuje její jedinečnost v rámci aplikace. Použití tohoto vzoru dále pomáhá optimalizovat výkon aplikace tím, že se minimalizuje počet operací, jako je vytváření nových instancí třídy nebo otevírání nových databázových spojení, které mohou zatěžovat systémové a síťové zdroje. [11]

```
// Singletone pro získávání stejné instance připojení do DB
Počet odkazů: 10
public static DatabaseConnector Instance
{
    get
    {
        databaseConnector ??= new DatabaseConnector();
        return databaseConnector;
    }
}
```

Obrázek 21: Struktura kódu – návrhový vzor Singleton [zdroj vlastní]

4 Technická dokumentace

4.1 Použité technologie

4.1.1 HTTP komunikace

HTTP je komunikační protokol užívaný pro vzájemnou komunikaci mezi webovým serverem a klientem, jako je webový prohlížeč či aplikace. Protokol HTTP funguje na principu dotazů a odpovědí: klient posílá HTTP požadavek (request) a server odpovídá (response).HTTP nezachovává mezi klientem a serverem trvalé spojení. Díky HTTP mohou uživatelé přistupovat k webovým stránkám dostupným na internetu. Šifrovaná varianta protokolu je označovaná jako HTTPS, která zajišťuje bezpečnější přenos dat mezi klientem a serverem. [12]

4.1.2 Asynchronizace

Asynchronní programování je vývojový přístup, který umožňuje efektivní provádění více úloh současně, aniž by došlo k zablokování programu čekáním na dokončení jednotlivých operací. Tento styl programování je široce využíván v oblastech jako je webový vývoj, síťová komunikace nebo vývoj grafických uživatelských rozhraní, a to díky jeho schopnosti udržet aplikace responzivní a efektivní. Klíčové prvky asynchronního programování jsou:

- **Asynchronní operace:** Program může pokračovat v dalších úlohách, zatímco jiná operace probíhá na pozadí, což zvyšuje celkovou efektivitu.
- **Callback funkce:** V asynchronním kódu se často využívají callback funkce, které se spustí po dokončení asynchronní operace.
- **Vlákna a paralelní programování:** I když asynchronní programování nemusí vždy zahrnovat více vláken, může zlepšit správu jednovláknového kódu tím, že umožňuje hlavnímu vláknu přepínat mezi úlohami. [13]

4.1.3 Jazyk C#

C# je moderní, objektově orientovaný programovací jazyk, který poskytuje vývojářům nástroje pro tvorbu různorodých, robustních a bezpečných aplikací běžících na platformě .NET. Jazyk, který je členem rodiny jazyků založených na C, má mnoho společných rysů s jazyky jako C++, Java nebo JavaScript. Jednou z klíčových vlastností C# je automatické spravování paměti, které je zajišťováno pomocí garbage collectoru, eliminujícího nepotřebné objekty z paměti. C# také podporuje asynchronní operace, lambda výrazy, nullable typy a obsluhu výjimek, což umožňuje tvorbu vysoce efektivního a optimalizovaného kódu. [14]

4.1.4 SQLite

SQLite je relační databázový systém založený na tabulkovém modelu, kde každá tabulka reprezentuje položky stejného typu. Jako netypovaná databáze SQLite nevyžaduje specifikaci datových typů sloupců, jako jsou celá čísla, reálná čísla, text nebo data, přesto je z hlediska správné praxe vhodné sloupce typovat. S databází se komunikuje prostřednictvím standardních SQL dotazů. Oproti větším databázím jako MySQL, MariaDB nebo PostgreSQL představuje SQLite pouze kompaktní knihovnu nástrojů, která je často již integrována přímo do některých programovacích jazyků. Každá databáze SQLite je uložena v jednom souboru na disku, což zajišťuje snadnou přenositelnost. Tento typ databáze je ideální pro menší až středně velké aplikace díky své jednoduchosti, avšak obvykle postrádá některé funkce, jako jsou uživatelská oprávnění, rozsáhlá konfigurace nebo plná podpora UTF znakové sady. [15]

4.1.5 API

API je rozhraní, které umožňuje komunikaci mezi dvěma nebo více softwarovými systémy. V kontextu integrace informačních systémů nebo webových aplikací API umožňuje standardizovaný způsob komunikace mezi rozdílnými systémy. Jednotlivé systémy mohou API implementovat pro nabízení specifických funkcí, zatímco jiné systémy tyto funkce využívají. Díky API mohou systémy komunikovat automaticky, bez lidského zásahu a nezávisle na programovacím jazyku. V oblasti webových či desktopových aplikací je pro implementaci API obvykle využíván HTTP protokol. Toto rozhraní pak funguje na principu zasílání HTTP požadavků na definované URL adresy, tzv. endpointy. Server zpracovává tyto požadavky a vrací odpovědi, které mohou být ve formátech jako JSON nebo XML, což jsou standardy pro výměnu dat mezi aplikacemi na internetu. [16]

4.1.6 WPF

WPF je pokročilá architektura pro tvorbu uživatelských rozhraní, která je optimalizována pro využití na různých typech displejů díky své nezávislosti na rozlišení a vektorovému vykreslovacímu modulu. Tento modul je navržen tak, aby co nejlépe využíval možnosti moderního grafického hardwaru. WPF poskytuje bohatou sadu nástrojů pro vývoj aplikací, včetně jazyka XAML pro definování uživatelského rozhraní, rozsáhlé knihovny ovládacích prvků, pokročilých možností pro datové vazby, podporu pro 2D a 3D grafiku, animace, styly, šablony, práci s textem a typografií, stejně jako integraci multimédií a dokumentů. Je součástí .NET frameworku a umožňuje vývojářům kreativně a efektivně kombinovat deklarativní značky a programový kód pro tvorbu dynamických a vizuálně atraktivních aplikací. [17]

4.1.7 .NET

.NET je bezplatná, open-source vývojářská platforma od Microsoftu, která umožňuje vytvářet různé typy aplikací. Podporuje mnoho programovacích jazyků, přičemž C# je nejpopulárnější. Platforma nabízí vysokou produktivitu, výkon, bezpečnost a spolehlivost, zahrnuje bohatou knihovnu funkcí optimalizovanou pro různé operační systémy a architektury. .NET používá automatickou správu paměti, podporuje souběžnost pomocí async/await a Task primitiv, a je navržen pro bezpečný kód s možností integrace nativního kódu. Platforma je pravidelně aktualizována a spravována jak Microsoftem, tak vývojářskou komunitou. .NET platforma zahrnuje několik klíčových komponent

- Modul runtime: Zajišťuje spuštění aplikací.
- Knihovny: Poskytují různé funkce, jako je analýza JSON.
- Kompilátor: Převádí zdrojový kód z jazyků jako C# do spustitelného formátu.
- SDK a další nástroje: Umožňují vývoj a sledování aplikací s využitím moderních technik.
- Zásobníky aplikací: Například ASP.NET Core a Windows Forms, které podporují vývoj aplikací. [18]

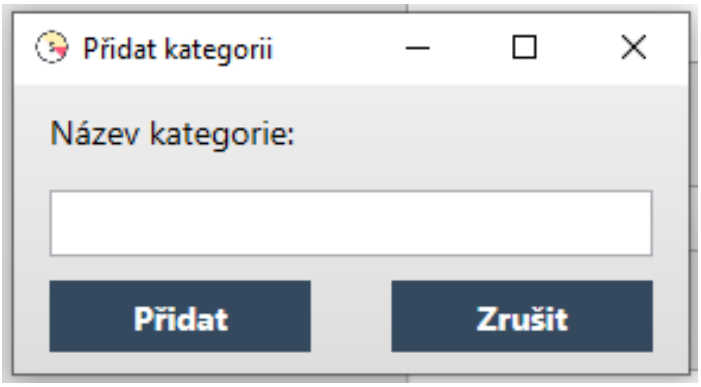
4.1.8 XAML

XAML je značkovací jazyk odvozený z XML, který se používá k deklarativnímu definování uživatelského rozhraní aplikací. Je běžně využíván pro tvorbu rozhraní různých komponent, jako jsou okna, dialogy, stránky a ovládací prvky. Díky své strukturované a intuitivní syntaxi umožňuje XAML vývojářům efektivně navrhovat vizuální aspekty aplikací, což usnadňuje oddělení vizuální prezentace od business logiky aplikace. Na následujícím obrázku je uveden příklad XAML kódu dialogu z této aplikace. [17]

```
<Window x:Class="FinanceTracker.Graphics.Dialogs.AddCategoryDialog" xmlns="http://schemas.microsoft.com/win
    <Grid Style="{StaticResource GradientBackgroundStyle}">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Label Content="Název kategorie:" Style="{StaticResource WindowsLabelStyle}"  Grid.Row="0"/>
        <TextBox x:Name="CategoryNameTextBox" Style="{StaticResource WindowsTextBoxStyle}"
                 Grid.Row="1" Margin="15,5,15,5" Width="NaN" HorizontalAlignment="Stretch"/>
        <Button Style="{StaticResource WindowsButtonStyle}" Grid.Row="2" Width="110"
                HorizontalAlignment="Left" Content="Přidat" Click="AddButton_Click"/>
        <Button Style="{StaticResource WindowsButtonStyle}" Grid.Row="2" Width="110"
                HorizontalAlignment="Right" Content="Zrušit" Click="CancelButton_Click" Margin="0,5,15,5"/>
    </Grid>
</Window>
```

Obrázek 22: Příklad XAML kódu [zdroj vlastní]

Kde výsledkem kódu je následující jednoduchý dialog umožňující přidání libovolné kategorie v modulu finance.



Obrázek 23: Dialog pro přidání kategorie [zdroj vlastní]

4.2 Knihovny

4.2.1 LiveChartsWPF

LiveCharts je zcela volně dostupná knihovna pro tvorbu grafů, kterou lze integrovat přímo do projektu pomocí balíčkovače Nuget. Umožňuje uživatelům tvorbu různorodých typů grafů, například sloupcové, koláčové, časové sběrnice atd. Tyto grafy lze rozšiřovat, podporují klasické typy zpracování událostí pomocí Event handlerů. Lze je stylizovat a mají integrovanou automatickou responzivitu. Data do grafů lze přenášet například pomocí datových vazeb (data bindings), to zaručuje automatické aktualizování informací zobrazovaných uživateli na základě změn datových modelů či zdrojů. V následujícím obrázku lze vidět tvorba grafu z knihovny a vázání dat. [19]

```
<lvc:CartesianChart x:Name="MonthlyCartChart" Series="{Binding MonthlyCartSeriesCollection}"
                    LegendLocation="Right" Margin="0,0,0,45" Visibility="Hidden">
    <lvc:CartesianChart.AxisX>
        <lvc:Axis Title="Kategorie" Labels="{Binding Labels}"></lvc:Axis>
    </lvc:CartesianChart.AxisX>

    <lvc:CartesianChart.AxisY>
        <lvc:Axis Title="Cena" LabelFormatter="{Binding Formatter}"></lvc:Axis>
    </lvc:CartesianChart.AxisY>
</lvc:CartesianChart>
```

Obrázek 24: Příklad tvorby grafu pomocí LiveCharts [zdroj vlastní]

4.2.2 Newtonsoft.Json

Newtonsoft.Json, známá také jako Json.NET, je volně dostupná knihovna, která poskytuje rozsáhlé možnosti pro práci s JSON daty v .NET aplikacích. Umožňuje efektivní serializaci objektů .NET do JSON formátu a naopak, deserilizaci JSON dat zpět na .NET objekty, což zjednodušuje práci s daty získanými z různých zdrojů, jako jsou například webové API. Knihovna lze snadno integrovat do projektů pomocí Nuget balíčkovače a je kompatibilní s mnoha typy .NET aplikací, včetně ASP.NET, WPF a Windows Forms. Díky své vysoké flexibilitě a široké podpoře se stala de facto standardem pro JSON operace v .NET ekosystému. V této aplikaci je specificky využita pro deserilizaci dat získávaných z API, což umožňuje snadné a rychlé zpracování dat bez nutnosti ručního parsingu JSON struktur. Její využití výrazně zvyšuje efektivitu a spolehlivost datových operací v aplikaci. [20]

4.2.3 System.Data.SQLite

System.Data.SQLite je knihovna umožňující připojení a manipulaci s SQLite databází přímo v rámci .NET. Poskytuje integraci SQLite databáze do .NET aplikací bez nutnosti složitého nastavení nebo externích databázových serverů. Je to ideální knihovna pro aplikace, které vyžadují lehkou databázovou komponentu s minimálními nároky na systémové zdroje. [21]

4.2.4 LINQ

Jedná se o sadu technologií, která umožňuje uživateli pomocí syntaxe dotazů provádět operace filtrování, řazení a seskupování zdrojů s minimem kódu. Výrazy syntaxí jsou podobné jazyku SQL. V následující části kódu je uveden příklad užití v aplikaci, kde se pomocí LINQ dotazu filtrují zobrazované kryptoměny v tabulce. [22]

```
// Filtruje kryptoměny podle zadaného textu
Počet odkazů: 1
private void FilterCryptoData(object sender, TextChangedEventArgs e)
{
    if (cryptoCurrencies == null)
    {
        return;
    }
    string filterText = CryptoFilterBox.Text;
    List<CryptoCurrency> filteredData = cryptoCurrencies.Where(crypto =>
        crypto.Name.Contains(filterText, StringComparison.OrdinalIgnoreCase) ||
        crypto.Symbol.Contains(filterText, StringComparison.OrdinalIgnoreCase) ||
        crypto.Rank.Contains(filterText, StringComparison.OrdinalIgnoreCase)).ToList();

    CryptoDataGrid.ItemsSource = filteredData;
}
```

Obrázek 25: Ukázka LINQ dotazu [zdroj vlastní]

4.2.5 CultureInfo

Knihovna CultureInfo je součástí .NET Frameworku, která umožňuje vývojářům efektivně řešit různé aspekty jazykové a kulturní lokalizace. Nabízí nástroje pro přizpůsobení aplikací specifikům různých kultur, jako je správné formátování čísel, měn, data a času, což je nezbytné pro mezinárodní aplikace. Využití této knihovny zajišťuje, že aplikace bude správně zobrazovat lokalizovaný obsah podle kulturních a jazykových preferencí uživatele, což přispívá k lepší uživatelské zkušenosti a zvyšuje globální použitelnost aplikace. [23]

4.3 Databázová vrstva aplikace

Pro uchovávání uživatelských informací byla zvolena malá databáze SQLite. Je přímo součástí projektu a nachází se v jeho výchozí složce – bin/debug/“verze .NET“, podsložka Data. Connection string, což je řetězec potřebný pro připojení k databázi se nachází v konfiguračním souboru aplikace. Při prvním spuštění aplikace vytvoří databázový soubor sqlite a databázovou strukturu s relačními vazbami pro správnou funkčnost aplikace.

4.3.1 Vkládání dat do databáze

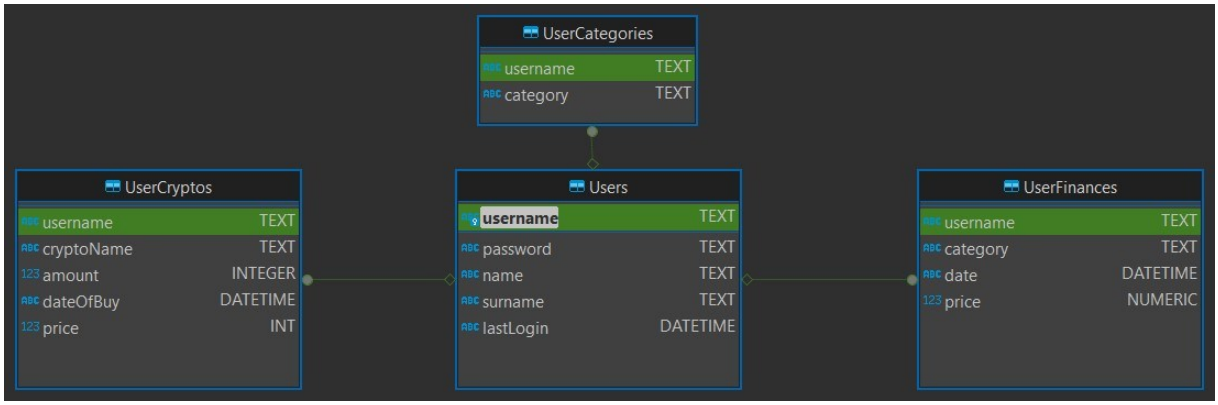
Data jsou do databáze předávána bezpečnou metodou pomocí parameterizovaných dotazů. Tento přístup zabraňuje rizikům spojeným s SQL injekcemi, což jsou běžné útoky, při kterých útočník vkládá nebo upravuje SQL příkazy s cílem ovlivnit databázové operace. Parameterizované dotazy fungují tak, že místo přímého vkládání uživatelských dat do SQL příkazu používají zástupné symboly (parametry), které jsou následně nahrazeny skutečnými hodnotami až v době vykonání dotazu. Tímto způsobem jsou data oddělena od kódu dotazu, což minimalizuje riziko neoprávněné manipulace s databází. Díky tomu je zajištěna vyšší bezpečnost a robustnost aplikace při zacházení s daty uživatelů. V následujícím obrázku je uveden příklad parametrizovaného dotazu registrace uživatele do databáze.

```
// Zajišťuje uživatelskou registraci do databáze
Počet odkazů: 1
public bool Register(string name, string surname, string username, string password)
{
    if (Util.UserExists(username))
    {
        Logger.WriteErrorLog(this, $"Uživatel se pokusil zaregistrovat s existujícím jménem '{username}'");
        Util.ShowErrorMessageBox("Uživatel s tímto uživatelským jménem již existuje, prosím, použijte jiné");
        return false;
    }
    string hashedPassword = Util.HashInput(password);
    string sql = "INSERT INTO Users Values(@username, @password, @name, @surname, @lastLogin)";
    using SQLiteCommand command = new(sql, connection);
    command.Parameters.AddWithValue("@username", username);
    command.Parameters.AddWithValue("@password", hashedPassword);
    command.Parameters.AddWithValue("@name", name);
    command.Parameters.AddWithValue("@surname", surname);
    command.Parameters.AddWithValue("@lastLogin", DateTime.Now);
    int success = command.ExecuteNonQuery();
```

Obrázek 26: Ukázka parametrizovaného dotazu [zdroj vlastní]

4.3.2 Relační model

Databáze se skládá ze 4 tabulek, UserCryptos, Users, UserFinances a UserCategories. Nachází se zde jediný primární klíč – username, musí být unikátní, toto je řešeno v aplikační stránce, to z něj dělá dokonalého kandidáta na primární klíč. Zbylé tabulky jsou propojené pomocí reference na tento sloupec.



Obrázek 27: Relační model databáze [zdroj vlastní]

4.3.3 Popis tabulek

UserCryptos – obsahuje informace o uživatelských kryptoměnách, tedy konkrétně těch, které evidoval v aplikaci jako nakoupené a tím je zanesl do sledování. Obsahuje referenci na username uživatele z tabulky Users, název kryptoměny, množství, které nakoupil, datum nákupu a nákupní cenu.

Users – tabulka evidovaných uživatelů, obsahuje jejich přihlašovací údaje, osobní informace a datum posledního přihlášení uživatele do aplikace

UserFinances – obsahuje informace o evidovaných uživatelských výdajích. Obsahuje reference na username uživatele z tabulky Users, kategorii nákupu, datum nákupu a cenu.

UserCategories – zde jsou evidované kategorie financí jednotlivých uživatelů které jsou používány pro zahrnutí uživatelsky evidovaných finanční do specifických odvětví.

5 Budoucí rozšíření

5.1 Profil

Celkové zvětšení užitku sekce profil, přidat například profilový obrázek či editaci uživatelských údajů pro zvýšení uživatelské přívětivosti aplikace.|

5.2 Finance

Šablony

Nabídka vytvořených šablon pro různé opakující se druhy výdajů, které by si mohli uživatelé snadno tvořit a upravovat. To by sloužilo k zjednodušení a zpříjemnění práce s aplikací.

Specifikace výdajů

Přidání pole pro popis dodatečných specifikací výdajů. Uživatel by si mohl k dané kategorii dopsat podrobnosti, například kategorie zdravotní péče, přidané specifikum léky, zubař či návštěva pohotovosti. Tato funkcionalita by zaručovala větší přehlednost v uživatelsky evidovaných záznamech a podrobnější analýzu pro samotné uživatele.

Editace evidovaných financí

Bylo by vhodné uživatele nechat editovat již evidované finance, aby v případě chyby nemusel daný záznam mazat a evidovat znovu.

Přidání jednotky měny

Nyní je aplikace navržena bez konkrétní měny při evidování uživatelských financí, přidání jednotky měny by přineslo vyšší uživatelskou přívětivost a pocit srozumitelnosti a relevance napříč uživateli z rozdílných zemí.

5.3 Převodník

Prohození měň

Možnost přehození zdrojové a cílové měny, což by sloužilo k optimálnějším a rychlejším výpočtům pro uživatele.

5.4 Kryptoměny

Upozornění

Nastavení upozornění na základě specifických tržních událostí nebo cenových hranic, které si uživatel předem definuje. Upozornění by dále definovalo, v jaké formě by jej uživatel chtěl přijímat, například prostřednictvím e-mailu či dialogového okna při výskytu dané události.

Souhrnný výpis

Přidání další sekce souhrnných výpisů, které by poskytli uživatelům komplexní přehled o stavu a vývoji jejich krypto portfolia. Například při vícenásobných investicích do stejných kryptoměn by souhrn ukazoval celkovou investovanou částku a celkové procentuální rozdíly vůči kupním cenám. Dále například přidání historie investic, která by dle všech vlastněných kryptoměn ukazovala uživatelsky evidovanou transakční historii.

5.5 Dashboard

Více typů grafů

Rozšíření nabídky grafů o nové typy, například bodové či spojnicové s rozšířenějšími možnostmi zobrazení, což by uživatelům umožnilo lepší vizualizaci a analýzu jejich dat.

Podrobnější legendy grafů

Toto by zlepšilo orientaci uživatelů v prezentovaných datech, což umožní rychlejší a přesnější interpretaci informací. Například přidání jednotky měny, ve které jsou finanční záznamy evidovány , do legendy by poskytlo lepší přehlednost daných záznamů, protože všechny záznamy nemusí být nutně evidovány ve stejných měnách, například útraty na dovolené v zahraničí.

Závěr

Cílem práce bylo vytvořit uživatelsky přívětivou aplikaci pro sledování osobních výdajů a investic, cíl této práce byl splněn.

Aplikaci bych zhodnotil jako velký přínos pro můj osobní rozvoj v rámci programovacích technik. Téma jsem si zvolil sám, protože mi přišlo zajímavé, zároveň je mi i blízké, protože se sám o investice a osobní finanční rozvoj zajímám.

Téma pro mě zároveň působilo jako výzva, protože s tvorbou desktopových aplikací nemám příliš mnoho zkušeností.

Seznámil jsem se s novými knihovnami a programovacími technologiemi a zároveň zlepšil své zkušenosti při práci s již pro mě známými principy.

Věřím, že aplikace může být přínosná pro různé spektrum uživatelů i v této formě bez doplňujících funkcionalit a modulů.

Použitá literatura

[1] Kryptoměna. *Wikipedia: the free encyclopedia*. San Francisco (CA): Wikimedia Foundation, 2014, 22.3.2024. Dostupné také z: https://cs.wikipedia.org/wiki/Kryptoměna

[2] Bitcoin. *CzechWealth* [online]. 2003 [cit. 2024-05-04]. Dostupné z: https://www.czechwealth.cz/slovnik-pojmu/bitcoin

[3] Asymetrická kryptografie. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2024-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Asymetrická_kryptografie

[4] Co je to token? *SubInvest* [online]. 2020 [cit. 2024-05-04]. Dostupné z: https://www.subinvest.cz/blog/co-je-to-token

[5] Co je blockchain, jak funguje a kde najde využití? *Rascasone* [online]. 2017 [cit. 2024-05-04]. Dostupné z: https://www.rascasone.com/cs/blog/zmeni-technologie-blockchain-cely-svet

[6] Volume. *LYNX* [online]. 2013 [cit. 2024-05-04]. Dostupné z: https://www.lynxbroker.cz/investovani/burzovni-trhy/burzovni-informace/investicni-slovnik/volume/

[7] Tržní kapitalizace. *LYNX* [online]. 2013 [cit. 2024-05-04]. Dostupné z: https://www.lynxbroker.cz/investovani/burzovni-trhy/burzovni-informace/investicni-slovnik/trzni-kapitalizace/

[8] SYROVÝ, Petr. *Investování pro začátečníky*. 4., zcela přepracované a rozšířené vydání. Praha: Grada Publishing, 2022. Investice. ISBN 978-80-271-3458-8.

[9] Frankfurter. *Frankfurter* [online]. 2018 [cit. 2024-04-08]. Dostupné z: https://www.frankfurter.app/docs/

[10] COINCAP. RESTful API documentation. *Coincap* [online]. 2015 [cit. 2024-04-08]. Dostupné z: https://docs.coincap.io/#ee30bea9-bb6b-469d-958a-d3e35d442d7a

[11] ITNETWORK. Lekce 2 - Singleton (jedináček). *Itnetwork.cz* [online]. 2015 [cit. 2024-04-19]. Dostupné z: https://www.itnetwork.cz/navrh/navrhove-vzory/gof/singleton-navrhovy-vzor

[12] STRELEC, Michal. Co je to HTTP při vývoji softwaru na míru. *Michal Strelec* [online]. 2018 [cit. 2024-05-04]. Dostupné z: https://www.strelec.pro/slovnik-vyvojare/co-je-to/http

[13] Asynchronní programování. *Expert Dev* [online]. 2019 [cit. 2024-05-04]. Dostupné z: https://www.expert-dev.cz/slovnicek-pojmu/asynchronni-programovani/

[14] SELLS, Chris. *C# a WinForms: programování formulářů Windows : Microsoft .NET development series*. Brno: Zoner Press, 2005. Encyklopedie Zoner Press. ISBN 80-86815-25-0.

[15] ITNETWORK. Lekce 1 - Úvod do SQLite a příprava prostředí. *Itnetwork.cz* [online]. 2015 [cit. 2024-04-17]. Dostupné z: https://www.itnetwork.cz/sqlite/sqlite-tutorial-uvod-a-priprava-prostredi

[16] STRELEC, Michal. Co je to API? *Michal Strelec* [online]. 2018 [cit. 2024-04-17]. Dostupné z: https://www.strelec.pro/napsal/co-je-to-api

[17] MICROSOFT. Průvodce pro desktop (WPF .NET). *Microsoft* [online]. 1975 [cit. 2024-04-17]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/desktop/wpf/overview/?view=netdesktop-8.0

[18] Úvod do technologie .NET. *Microsoft* [online]. 1975 [cit. 2024-05-05]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/core/introduction

[19] LIVECHARTS. Tutorial and Samples. *LiveCharts* [online]. 2004 [cit. 2024-04-19]. Dostupné z: https://v0.lvcharts.com/App/examples/Wpf/start

[20] JSON.NET. Serializing and Deserializing JSON. *Newtonsoft* [online]. 2014 [cit. 2024-04-19]. Dostupné z: https://www.newtonsoft.com/json/help/html/SerializingJSON.htm

[21] MICROSOFT. Porovnání se System.Data.SQLite. *Microsoft* [online]. 1975 [cit. 2024-04-19]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/standard/data/sqlite/compare

[22] MICROSOFT. LINQ (Language Integrated Query). *Microsoft* [online]. 1975 [cit. 2024-04-16]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/linq/

[23] CultureInfo Třída. *Microsoft* [online]. 1975 [cit. 2024-05-05]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/api/system.globalization.cultureinfo?view=net-8.0