EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103003/B/2025/36146475401641732

KONŠTRUKCIA WEBOVEJ EDUKAČNEJ POMÔCKY PRI INVESTOVANÍ VO FRAMEWORKU

Bakalárska práca

2025 **Matúš Kalník**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

KONŠTRUKCIA WEBOVEJ EDUKAČNEJ POMÔCKY PRI INVESTOVANÍ VO FRAMEWORKU

Bakalárska práca

Študijný program: Hospodárska Informatika
Študijný odbor: Ekonómia a manažment
Školiace pracovisko: Katedra operačného výskumu a ekonometrie
Vedúci záverečnej práce: Ing. Richard Martinus

Bratislava 2025 **Matúš Kalník**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Matúš Kalník
Študijný program: hospodárska informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: ekonómia a manažment
Typ záverečnej práce: Bakalárska záverečná práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Konštrukcia webovej edukačnej pomôcky pri investovaní vo frameworku

Anotácia: Bakalárska práca sa venuje konštrukcii webovej edukačnej pomôcky zameranej na investovanie, ktorá je vyvinutá vo vybranom webovom frameworku. Práca analyzuje súčasné edukačné nástroje v oblasti investovania, identifikuje ich silné a slabé stránky, a na základe týchto poznatkov navrhuje vlastnú interaktívnu platformu. Cieľom je poskytnúť používateľom komplexný, intuitívny a prístupný nástroj, ktorý podporuje pochopenie základných aj pokročilých investičných stratégií a konceptov. Práca obsahuje návrh užívateľského rozhrania, implementáciu edukačných modulov a ich integráciu do frameworku, ako aj hodnotenie efektivity nástroja v edukačnom procese.

Vedúci: Ing. Richard Martinus
Katedra: KOVE FHI - Katedra operačného výskumu a ekonometrie

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 09.03.2024

Dátum schválenia: 20.03.2024

doc. Ing. Martin Mišút, CSc.
osoba zodpovedná za realizáciu študijného programu

Čestné vyhlásenie

Čestne vyhlasujem, že som svoju bakalársku prácu s názvom „*Konštrukcia webovej edukačnej pomôcky pri investovaní vo frameworku“* vypracoval samostatne a v texte práce som náležite uviedol a citoval všetku použitú literatúru a zdroje.

Dátum: .....

podpis študenta

Poďakovanie

Moja úprimná vďaka patrí môjmu školiteľovi, Ing. Richardovi Martinusovi, za jeho odborné vedenie, cenné rady a trpezlivý prístup počas vypracovania tejto bakalárskej práce.

ABSTRAKT

KALNÍK, Matúš: *Konštrukcia webovej edukačnej pomôcky pri investovaní vo frameworku.* – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra operačného výskumu a ekonometrie. – Vedúci záverečnej práce: Ing. Richard Martinus. Bratislava: FHI, 2025, 59 strán.

Cieľom bakalárskej práce je prepojiť teóriu s praxou prostredníctvom webovej aplikácie, ktorá používateľom uľahčí pochopenie investičných princípov. Aplikácia umožňuje užívateľovi zostaviť si vlastné portfólio z databázy akcií a ETF fondov. Prvá časť práce je zameraná na teoretické a technologické základy, ktoré slúžia ako základ pre návrh a vývoj aplikácie. V druhej časti sú definované ciele a metodika práce. Praktická časť popisuje využitie jednotlivých funkcií. Výsledkom práce je jednoducho ovládateľná webová aplikácia postavená na frameworku Flask a jazyku Python. Aplikácia poskytuje praktický pohľad na fungovanie investovania bez potreby reálneho kapitálu. Môže slúžiť ako učebná pomôcka pre študentov, samoukov, ale aj pre širšiu verejnosť. Je navrhnutá tak, aby ju bolo možné v budúcnosti jednoducho rozšíriť o ďalšie funkcionality.

Kľúčové slová: investovanie, webová aplikácia, tvorba portfólia, Flask, vizualizácia dát

ABSTRACT

KALNÍK, Matúš: *Construction of a Web-Based Educational Tool for Investing Using a Framework*. – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Operations Research and Econometrics. – Thesis supervisor: Ing. Richard Martinus. Bratislava: FHI, 2025, 59 pages.

The aim of the bachelor's thesis is to connect theory with practice through a web application that will make it easier for users to understand investment principles. The application allows the user to build their own portfolio from a database of stocks and ETF funds. The first part of the thesis focuses on the theoretical and technological foundations that serve as the basis for the design and development of the application. The second part defines the goals and methodology of the work. The practical part describes the use of individual functions. The result of the thesis is an easy-to-use web application built on the Flask framework and the Python language. The application provides a practical view of how investing works without the need for real capital. It can serve as a teaching aid for students, self-taught people, but also for the wider public. It is designed so that it can be easily expanded with additional functionalities in the future.

Keywords: investing, web application, portfolio creation, Flask, data visualization

Obsah

Úvod

Investovanie je forma nakladania s financiami, pri ktorej sa človek snaží zhodnotiť svoje finančné prostriedky tým, že ich vloží do rôznych aktív ako sú napríklad akcie, dlhopisy či ETF fondy. Aby však vedel investovať efektívne, musí rozumieť viacerým faktorom, medzi ktoré patrí fungovanie trhov, riziká spojené s jednotlivými aktívami a význam diverzifikácie portfólia. Nie každý má možnosť si tieto veci vyskúšať „nanečisto“, a práve preto vznikla myšlienka vytvoriť webovú edukačnú aplikáciu, ktorá to umožňuje.

V aplikácii sa bude pracovať s technológiami Flask, Python a SQLAlchemy, ktoré spolu zabezpečia spracovanie údajov, prácu s databázou a tvorbu interaktívneho webového rozhrania. Používateľ si bude môcť jednoducho vybrať akcie alebo ETF fondy z databázy, nastaviť im váhové zastúpenie a sledovať, ako by sa jeho portfólio vyvíjalo. Okrem výpočtu alokácií aplikácia zobrazí aj základné investičné metriky ako výnosy, volatilitu či sektorové a geografické rozloženie. Dáta sa budú získavať z reálneho trhu pomocou knižnice yFinance.

V prvej časti práce sa vysvetlia teoretické pojmy ako investičné portfólio, diverzifikácia, volatilita alebo výnosy. Tie budú slúžiť ako základ na pochopenie toho, čo bude aplikácia vlastne simulovať. Následne sa opíše technická architektúra aplikácie, jednotlivé použité knižnice a spôsob ich implementácie.

V praktickej časti sa budú riešiť výpočty a zobrazovanie výsledkov tak, aby mal používateľ čo najlepší prehľad. Cieľom bude vytvoriť nástroj, ktorý pomôže pochopiť investovanie aj tým, čo s ním len začínajú. Všetko sa navrhne jednoducho a zrozumiteľne.

V záverečnej časti práce sa zhodnotia silné a slabé stránky vytvorenej webstránky. Medzi hlavné výhody aplikácie bude patriť jej používateľská jednoduchosť, prehľadné rozhranie a možnosť práce s aktuálnymi trhovými dátami. Určité obmedzenia sa môžu prejaviť napríklad pri presnosti historických údajov, alebo v nedostupnosti pokročilejších analytických nástrojov, ako je optimalizácia portfólia na základe Sharpeho pomeru. Aj napriek tomu bude môcť aplikácia slúžiť ako výborný výučbový nástroj pre študentov, samoukov, ale aj pre širšiu verejnosť, ktorá si bude chcieť vytvoriť základný prehľad v oblasti investovania bez rizika straty reálnych peňazí.

1 Súčasný stav riešenej problematiky

Investovanie sa dnes stáva obľúbeným spôsobom, ako si ľudia zabezpečujú finančnú budúcnosť pre seba či svoju rodinu. Vďaka internetu dnes môže investovať prakticky každý aj bez predošlých skúseností. Napriek tomu je pre začiatočníkov investovanie stále neznámou oblasťou. Dôvodom je najmä to, že téme investovania sa v tradičnom vzdelávacom systéme venuje len veľmi málo pozornosti, a preto aj študenti, ktorí sa s ňou stretávajú na vysokej škole, často pristupujú k tejto oblasti ako začiatočníci.

Na trhu existuje množstvo teoretických zdrojov, článkov, videí či online kurzov zameraných na vysvetlenie základov investovania. Problém však spočíva v ich pasívnom charaktere – používateľ väčšinou iba prijíma informácie bez možnosti ich okamžitého uplatnenia.

1.1 Nástroje na simuláciu a učenie

Na trhu existujú nástroje, ktoré umožňujú používateľom „vyskúšať si investovanie“ bez rizika. Medzi najznámejšie patria:

- **Investopedia Simulator** – interaktívny nástroj, ktorý simuluje obchodovanie na burze s použitím virtuálnych peňazí. Je vhodný na oboznámenie sa so základnými princípmi nákupu a predaja akcií, no v niektorých prípadoch pôsobí zastaralo a jeho používateľské rozhranie nie je úplne intuitívne pre začiatočníkov.
- **TradingView** – pokročilá platforma na technickú analýzu s možnosťou vizualizácie historických dát a vytvárania vlastných indikátorov. Výhodou je dostupnosť kvalitných grafov a veľká komunita, ktorá zdieľa svoje analýzy. Nevýhodou je však, že platforma je zameraná skôr na technicky zdatnejších používateľov – pre začiatočníka môže byť príliš komplexná a zahlcujúca.
- **Demo účty u brokerov** – mnohí poskytovatelia ako eToro, XTB alebo Trading 212 ponúkajú demo kontá s fiktívnym kapitálom. Tieto platformy však často smerujú používateľa skôr k reálnemu obchodovaniu a menej sa venujú edukačnému aspektu. Chýba im vysvetlenie kľúčových metrík či vizualizácia dlhodobého vývoja portfólia.

1.2 Slabé stránky existujúcich riešení

Väčšina dnes dostupných investičných platforiem má jednu vec spoločnú. Často sú buď príliš technicky zamerané, alebo málo prístupné z pohľadu bežného používateľa. Typickým príkladom je TradingView, ktoré ponúka množstvo pokročilých nástrojov ako technické indikátory, sviečkové grafy či analýzu výkonnosti na dennej báze. Hoci ide o veľmi silný nástroj pre pokročilých obchodníkov, pre študenta alebo niekoho, kto s investovaním iba začína, môže byť celé rozhranie prehustené informáciami a ťažko uchopiteľné. Namiesto pochopenia základov investovania sa tak používateľ ľahko stratí v technických detailoch, ktoré mu v úvode nemusia dávať praktický zmysel. Takýto používateľ sa môže namiesto pochopenia významu diverzifikácie rýchlo stratiť v zložitých technických detailoch. Okrem toho sú mnohé aplikácie komerčne uzavreté. Jedná sa o platené riešenia, ktoré neumožňujú ďalší vývoj, úpravu obsahu či integráciu vlastných scenárov. V prípade výučby, kde je potrebné simulovať konkrétne prípady napríklad zostavenie portfólia pre slovenského študenta s obmedzeným rozpočtom – to predstavuje značnú prekážku.

Jednou z častých slabín týchto platforiem je aj ich pasívny charakter. Používateľ sa väčšinou len pozerá na grafy, číta pripravené články alebo sleduje preddefinované dáta, no nemá možnosť si niečo sám vyskúšať, overiť alebo aktívne zasiahnuť do simulácie. Chýba priestor na vytváranie vlastných hypotéz a ich testovanie v kontrolovanom prostredí, čo by mohlo viesť k lepšiemu pochopeniu súvislostí. Takýto pasívny prístup ide proti súčasným trendom vo vzdelávaní, ktoré kladú dôraz na zapojenie používateľa a praktické skúsenosti.

Platforma TradingView síce ponúka širokú škálu funkcií a nástrojov, no pri prvom kontakte s ňou sa veľká časť nových používateľov cíti zahltená. Zložité rozhranie, množstvo grafov a technických indikátorov často pôsobí demotivačne a podľa dostupných údajov až 70 % začiatočníkov po prvom kontakte stráca záujem a k platforme sa už nevracia. Ďalším problémom je obmedzená zákaznícka podpora – bezplatní používatelia nemajú prístup k priamej pomoci a na odpoveď cez e-mail môžu čakať aj tri až päť dní, čo je obzvlášť problematické počas kritických období (CANVAS BUSINESS MODEL, 2025).

Investopedia Simulator je obľúbený najmä pre svoju edukačnú hodnotu, no aj on má svoje limity. Jedným z nich je oneskorené zobrazovanie trhových dát – v prípade tohto

simulátora ide zvyčajne o meškanie približne 15 minút. Pri testovaní dlhodobejších stratégií to nemusí predstavovať problém, avšak pri snahe reagovať na aktuálne pohyby na trhu alebo pri krátkodobých stratégiách môže dochádzať k skresleniu výsledkov. Okrem toho je výber cenných papierov na platforme značne obmedzený. Niektoré zahraničné akcie alebo nízko ocenené tituly (tzv. penny stocks) nie sú dostupné, čo môže negatívne ovplyvniť diverzifikáciu a realistickosť portfólia (SMITH Lisa, 2025).

1.3 MAPS: Systém riadenia portfólia založený na multiagentnom posilňovacom učení

Systém MAPS predstavuje moderný a inovatívny prístup k riadeniu investičného portfólia, ktorý využíva spoluprácu viacerých agentov fungujúcich na princípe posilňovacieho učenia. Každý z týchto agentov vystupuje ako samostatný „investor“, ktorý sa rozhoduje nezávisle a vytvára vlastné portfólio. Výsledkom je diverzifikované a efektívne portfólio, ktorého cieľom je dosahovať čo najvyšší výnos pri súčasnom obmedzení rizika. Na rozdiel od tradičných metód, ako je napríklad Markowitzova teória optimálneho portfólia, ktoré pracujú so statickými predpokladmi o výnosoch a rizikách, MAPS reaguje dynamicky na vývoj trhu. Využíva spätnú väzbu zo svojho „prostredia“ a na jej základe priebežne upravuje investičné rozhodnutia. Takýto prístup umožňuje väčšiu flexibilitu a lepšiu adaptáciu na meniace sa podmienky finančných trhov (Lee et al., 2020).

Experimenty realizované na 12-ročných historických údajoch z amerického trhu potvrdzujú, že systém MAPS dosahuje vyšší Sharpeho pomer v porovnaní s klasickými prístupmi. Tento výsledok naznačuje, že dokáže ponúknuť lepší pomer medzi výnosom a rizikom. Zaujímavým zistením je aj to, že zapojením väčšieho počtu agentov do systému sa zvyšuje diverzifikácia a tým aj celková výkonnosť portfólia. MAPS tak predstavuje perspektívnu technológiu, ktorá môže nájsť uplatnenie v praxi ako nástroj pre investorov či analytikov, ktorí hľadajú adaptívne a inteligentné riešenia pri správe investičného portfólia.

2 Cieľ práce

Primárne ciele tejto bakalárskej práce sú návrh a vývoj interaktívnej webovej aplikácie, ktorá bude slúžiť ako edukačný nástroj v oblasti investovania do akcií a ETF. Aplikácia má prispieť k zvýšeniu finančnej gramotnosti používateľov, a to najmä prostredníctvom praktickej simulácie správy a tvorby vlastného investičného portfólia.

Samotná aplikácia, ktorá je výsledkom tejto bakalárskej práce, bude poskytovať viaceré funkcionality zamerané na praktické modelovanie investovania. V prvom rade umožní používateľovi zostaviť si vlastné investičné portfólio. Výber prebieha z databázy akcií a ETF fondov, ktorá bola vytvorená na základe údajov zo súboru pripraveného v prostredí Microsoft Excel. Ku každému zvolenému titulu môže používateľ priradiť percentuálnu alokáciu v rámci investičného rozpočtu vo výške 10 000 €.

Ďalším dôležitým pilierom aplikácie je práca s trhovými údajmi. Tie budú načítavané z verejne dostupných zdrojov, konkrétne z Yahoo Finance, a budú slúžiť ako základ na výpočet dôležitých investičných ukazovateľov – ročného výnosu (YTD), volatility, priebežnej hodnoty portfólia, ako aj celkového zisku alebo straty. Súčasťou riešenia bude aj vizualizácia týchto údajov, aby mal používateľ lepší prehľad o vývoji svojho portfólia. Priebežný vývoj bude zobrazený pomocou čiarových grafov, zatiaľ čo rozloženie portfólia z pohľadu sektorov a krajín bude ilustrované prostredníctvom koláčových grafov.

Neoddeliteľnou súčasťou aplikácie bude aj správa používateľských účtov. Po registrácii a prihlásení si bude môcť každý používateľ vytvoriť vlastné portfólio, ktoré sa uloží do databázy a bude dostupné na ďalšiu aktualizáciu. Okrem toho bude mať možnosť meniť si prihlasovacie údaje ako meno či heslo. Výsledkom bakalárskej práce bude teda plne funkčná webová aplikácia, ktorá poskytne nástroje na zostavenie investičného portfólia z akcií a ETF, vypočíta základné metriky výkonnosti a umožní ich vizuálnu prezentáciu v čase.

3 Metodika práce a metódy skúmania

V rámci vývoja webovej aplikácie je potrebné vždy zvážiť, ktoré technológie vyhovujú požiadavkám na jednoduchý vývoj, rýchlosť a flexibilitu, ale zároveň dokážu efektívne narábať s dátami. V nasledujúcich kapitolách budú opísané tieto technológie a nástroje, ktoré boli použité počas tvorby tejto práce.

3.1 Investičné portfólio a jeho význam

Investičné portfólio predstavuje súbor finančných aktív, ktoré investor drží s cieľom zhodnotiť svoj kapitál. Môžu to byť akcie, dlhopisy, ETF fondy, komodity alebo hotovosť. Správna tvorba portfólia závisí od investičných cieľov, časového horizontu a taktiež aj od ochoty prijať riziko. Dôležitá je do istej miery aj aktuálna situácia na finančných trhoch. Dosiahnutie optimálneho pomeru medzi očakávaným výnosom a určitou akceptovateľnou mierou rizika je kľúčovým cieľom tvorby každého jedného portfólia (TARDI Carla, 2024).

Jednou zo základných stratégií pri tvorbe portfólia je diverzifikácia. Spočíva v rozložení všetkých investícií medzi viaceré typy aktív alebo do viacerých sektorov. Znižuje sa tým pádom vplyv negatívneho vývoja jedného konkrétneho aktíva na celkový výkon daného portfólia. Ako uvádza Reilly & Brown (2012), diverzifikácia pomáha eliminovať špecifické riziká spojené s jednotlivými cennými papiermi, čím zostáva v portfóliu najmä systematické riziko, ktoré je neodstrániteľné.

V aplikácii vytvorenej v rámci tejto bakalárskej práce sa portfólio skladá z používateľom zvolených akcií a ETF fondov. Používateľ priraďuje jednotlivým titulom percentuálnu alokáciu, pričom systém následne prepočíta investované sumy a sleduje vývoj portfólia v čase. Tento proces simuluje reálne investičné správanie a umožňuje používateľovi v náučnej forme pochopiť princípy rozloženia rizika a výkonnosti portfólia.

3.2 Programovací jazyk Python

Python je moderný a vysoko čitateľný programovací jazyk s dynamickou typovou kontrolou, ktorý sa vyznačuje svojou rozsiahlou využiteľnosťou, prehľadnosťou zápisu a

jednoduchosťou. Používa sa na tvorbu webových aplikácií, analýzu dát, automatizáciu procesov alebo vývoj umelej inteligencie v komerčnej sfére a zaraďuje sa medzi najobľúbenejšie jazyky taktiež vo vedeckovýskumnej oblasti. V oblasti webového vývoja sa uplatňuje čoraz viac. Je to zapríčinené širokou škálou frameworkov, obsiahlymi dokumentáciami a aktívnou komunitou (Python Software Foundation, 2025).

V tejto bakalárskej práci je Python, za účelom vytvorenia logiky aplikácie a spracovania údajov o akciách, zvolený ako hlavný programovací jazyk. Výhodou Pythonu je, že sa s ním pracuje ľahko vďaka prehľadnej syntaxi a množstvu dostupných knižníc, ktoré vývoj výrazne urýchľujú. V rámci projektu sa využívali najmä knižnice ako pandas, ktorá slúži na prácu s tabuľkovými údajmi, yfinance na získavanie aktuálnych aj historických cien akcií a ETF, a json na prevod dát do formátu vhodného na ukladanie alebo prenos. Vďaka týmto nástrojom bolo možné jednoducho načítať dáta, vyhodnotiť výkonnosť portfólia a zobrazovať výsledky takmer okamžite.

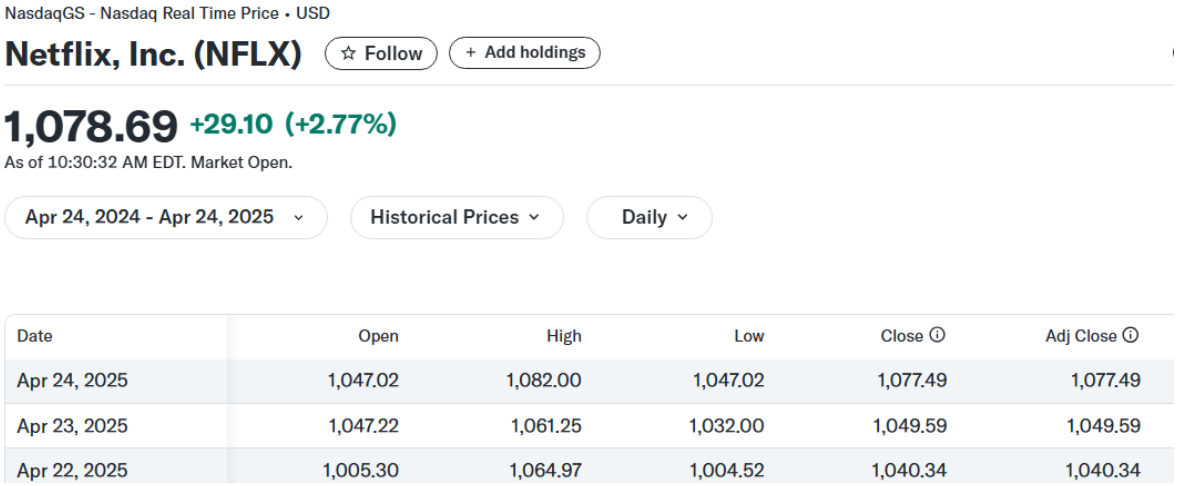
3.2.1 Knižnica yFinance

Knižnica yFinance je open-source knižnica pre jazyk Python, ktorá slúži na jednoduché získavanie historických a aktuálnych finančných údajov priamo z portálu Yahoo Finance. Vďaka tejto knižnici je možné programovo pristupovať k dátam o akciách, ETF fondoch a iných investičných nástrojoch bez nutnosti manuálneho exportu.

Z praktického hľadiska je yFinance veľmi populárna najmä pre svoj jednoduchý zápis, široký rozsah dostupných údajov a pravidelné aktualizácie. Umožňuje napríklad načítanie zatváracích cien akcií, objemov obchodovania, základných informácií o spoločnosti (ako názov, sektor či krajina), ako aj výpočet výnosov za určité obdobia (Ran, Aroussi, 2025).

V tejto bakalárskej práci bola knižnica yFinance použitá na dynamické získavanie aktuálnych a historických cien akcií a ETF. Tieto údaje sa následne využívali pri výpočtoch výkonnosti portfólia, YTD výnosov a volatility. Ako je vidieť na obrázku 1, cena "Adjusted Close" (upravená cena) je zobrazená v tabuľke historických cien. Táto cena je upravená o dividendy a rozdelenie akcií, čo poskytuje presnejší obraz o skutočnom vývoji ceny akcie v čase. Tento údaj je rovnaký ako ten, ktorý sa bude získavať cez API v Pythone pomocou knižnice yFinance, čím sa zabezpečí konzistencia údajov v aplikácii.

Obrázok 1: Zobrazenie historických cien akcií spoločnosti Netflix na portáli Yahoo Finance



NasdaqGS - Nasdaq Real Time Price • USD

Netflix, Inc. (NFLX) ☆ Follow + Add holdings

1,078.69 +29.10 (+2.77%)

As of 10:30:32 AM EDT. Market Open.

Apr 24, 2024 - Apr 24, 2025 | Historical Prices | Daily

Date	Open	High	Low	Close	Adj Close
Apr 24, 2025	1,047.02	1,082.00	1,047.02	1,077.49	1,077.49
Apr 23, 2025	1,047.22	1,061.25	1,032.00	1,049.59	1,049.59
Apr 22, 2025	1,005.30	1,064.97	1,004.52	1,040.34	1,040.34

Zdroj: Yahoo Finance

3.2.2 Knižnica Pandas

Pandas je obľúbená open-source knižnica pre jazyk Python, ktorá výrazne uľahčuje prácu so štruktúrovanými dátami. Najviac sa hodí pri analýze tabuliek alebo časových radov – teda údajov, ktoré sú časovo usporiadané. Vďaka svojej flexibilite si našla miesto vo viacerých oblastiach, ako sú financie, ekonómia, strojové učenie či vizualizácia dát. Jej základným stavebným prvkom je DataFrame – dvojrozmerná dátová štruktúra, ktorá funguje podobne ako tabuľka v Exceli, no ponúka omnoho viac možností, ako s údajmi pracovať efektívne a rýchlo (McKinney Wes, 2022).

V rámci tejto bakalárskej práce bola knižnica Pandas použitá na načítanie a úpravu údajov zo súboru stocks.xlsx, ktorý obsahoval informácie ako ticker, názov spoločnosti, sektor a krajinu. Dáta boli následne spracované a uložené do formátu DataFrame, ktorý sa ďalej využíval pri výbere akcií a ETF fondov do investičného portfólia. Medzi užitočné funkcie patrili napríklad *read_excel()* na import údajov, *iterrows()* na prechádzanie jednotlivých riadkov a *tolist()* na konverziu stĺpcov do zoznamov. Vďaka tejto knižnici bolo možné efektívne pripraviť dáta na ďalšiu analýzu a vizualizáciu v prostredí webovej aplikácie.

3.2.3 Knižnica Plotly

Plotly predstavuje moderný nástroj na vizualizáciu údajov v jazyku Python. Slúži na tvorbu vizuálne pútavých a interaktívnych grafov. Je navrhnutá tak, aby sa dala jednoducho integrovať do webových aplikácií. V porovnaní so statickými nástrojmi, ako je napríklad Matplotlib, umožňuje užívateľovi priamo pracovať s grafmi – približovať si údaje, posúvať zobrazenie, zobraziť presné hodnoty alebo exportovať vizualizáciu vo formáte HTML.

Vďaka svojim širokým možnostiam je Plotly často používaný vo finančných, vedeckých aj biznis aplikáciách, kde je dôležité prehľadné a zrozumiteľné zobrazenie veľkého množstva údajov. Medzi najčastejšie využívané typy vizualizácií patria čiarové grafy, koláčové grafy, histogramy, rozptylové diagramy či kombinované viacvrstvové grafy (Plotly Technologies Inc., 2025).

V rámci tejto bakalárskej práci sa knižnica Plotly využívala najmä na zobrazenie grafov vývoja hodnoty investičného portfólia, na porovnanie výkonnosti jednotlivých akcií, ako aj na vizualizáciu sektorového a geografického rozdelenia investícií. Grafy boli generované vo forme HTML fragmentov, ktoré sa následne zobrazovali priamo v používateľskom rozhraní aplikácie. Týmto spôsobom sa podarilo vytvoriť dynamické a prehľadné vizualizácie, ktoré používateľom umožňovali lepšie pochopiť výkonnosť ich portfólia.

3.3 Webový framework Flask

V tejto bakalárskej práci Flask zabezpečuje celú serverovú časť aplikácie. Rieši prihlasovanie používateľov, správu portfólií, prácu s databázou, vytváranie HTML stránok a zobrazenie grafov. S pomocou rozšírení ako Flask-Login, Flask-WTF alebo Flask-Migrate sa ľahko pridali funkcie ako bezpečné prihlásenie, kontrola údajov vo formulároch a správa databázy.

Flask je minimalistický mikroframework pre jazyk Python, ktorý slúži na vývoj webových aplikácií. Bol navrhnutý s dôrazom na jednoduchosť, flexibilitu a rozšíriteľnosť, vďaka čomu patrí medzi najobľúbenejšie riešenia pre rýchly vývoj menších alebo stredne veľkých projektov. Na rozdiel od robustnejších frameworkov, ako je napríklad Django, Flask neobsahuje rozsiahlu funkcionalitu už v základnej inštalácii. Namiesto toho ponecháva

vývojárovi voľnosť prispôsobiť si aplikáciu podľa vlastných potrieb výberom iba tých komponentov, ktoré sú pre daný projekt nevyhnutné (Flask, 2024).

Základom každej Flask aplikácie sú tzv. routy – cesty, ktoré definujú odpovede na konkrétne HTTP požiadavky. Okrem nich framework poskytuje aj šablónovací systém Jinja2, podporu pre spracovanie formulárov, cookies, session premenných či prácu s JSON formátom.

3.4 Správa databázy pomocou SQLAlchemy

SQLAlchemy je jedným z najvýznamnejších a najrozšírenejších nástrojov na prácu s databázami v prostredí programovacieho jazyka Python. Namiesto toho, aby sa pracovalo s databázou priamo pomocou SQL príkazov, SQLAlchemy poskytuje výkonné ORM (z angl. Object-Relational Mapping) rozhranie, ktoré umožňuje prácu s databázovými tabuľkami vo forme Python objektov.

Použitím SQLAlchemy sa vytvárajú tzv. modely, ktoré predstavujú tabuľky v databáze. Každý záznam (riadok) je potom reprezentovaný ako inštancia triedy. Táto forma abstrakcie výrazne zjednodušuje manipuláciu s dátami, vkladaním, aktualizáciou alebo odstraňovaním záznamov bez nutnosti písať priamo SQL kód. (SQLAlchemy, 2025).

V rámci tejto bakalárskej práce sa SQLAlchemy využíva na vytváranie štruktúry databázy, ktorá obsahuje používateľské účty, investičné portfóliá a ich historické dáta. ORM vrstva zabezpečuje efektívne prepojenie medzi backendovou časťou aplikácie (naprogramovanou vo Flasku) a relačnou databázou.

3.5 Frontend framework Bootstrap

Bootstrap je populárny open-source framework na tvorbu responzívnych a vizuálne konzistentných webových rozhraní. Vyvinutý bol pôvodne spoločnosťou Twitter a od svojho vzniku si získal obľubu medzi vývojármi pre svoju jednoduchosť, predpripravené komponenty a flexibilitu pri tvorbe front-endu.

Bootstrap 5, aktuálna verzia, ponúka vylepšenú podporu pre moderné prehliadače, odstránenie závislosti na jQuery a prechod na využitie vlastných JavaScript modulov. Využíva systém mriežky (z angl. grid system) na tvorbu rozvrhnutia stránky. Obsahuje

množstvo preddefinovaných štýlov a prvkov ako sú tlačidlá, formuláre, navigačné panely alebo tabulky čo výrazne urýchľuje vývoj (Bootstrap, 2024).

V rámci tejto bakalárskej práce bol Bootstrap použitý na tvorbu responzívneho dizajnu aplikácie, zabezpečenie kompatibility medzi zariadeniami (počítače, tablety, mobily) a na jednotný vzhľad používateľského rozhrania. Uplatnil sa najmä pri návrhu navigačného panela, formulárov pre zadávanie údajov, stránkovania a zobrazenia tabuliek s finančnými titulmi.

3.6 Vývojové prostredie Visual Studio Code

Visual Studio Code (VS Code) je moderné a rozšírené open-source vývojové prostredie vytvorené spoločnosťou Microsoft. Poskytuje široké možnosti úprav a rozšírení vďaka množstvu dostupných rozšírení, ktoré podporujú rôzne jazyky, frameworky a nástroje.

Tento editor bol zvolený ako hlavné vývojové prostredie pre túto bakalársku prácu najmä z dôvodu jeho flexibility, intuitívneho používateľského rozhrania a bezproblémovej integrácie s technológiami použitými v projekte, ako sú Python, HTML, CSS, JavaScript či SQLAlchemy. VS Code ponúka pokročilé funkcie ako automatické dopĺňanie kódu, zvýrazňovanie syntaxe, vstavaný terminál, Git integráciu a debugger, čo výrazne prispieva k efektivite vývoja (Visual Studio Code, 2024).

Pri tvorbe webovej aplikácie umožnil Visual Studio Code rýchle prepínanie medzi jednotlivými súbormi backendu, frontendu aj databázových modelov. Možnosť okamžitého spúšťania Flask servera priamo z editoru a jednoduché ladenie v prostredí rozšírenia „Python“ boli kľúčové pre plynulý priebeh implementácie.

3.7 Architektúra webovej aplikácie a REST API

Webová aplikácia vytvorená v rámci tejto bakalárskej práce je postavená na štandardnej architektúre typu klient-server. Základná logika a spracovanie dát prebieha na strane servera, ktorý je vytvorený pomocou frameworku Flask. Klientská časť, ktorá sa vykresľuje v prehliadači používateľa, využíva HTML šablóny s komponentmi Bootstrapu pre vizuálne zobrazenie údajov a interakciu.

Komunikácia medzi používateľom (klientom) a serverom prebieha prostredníctvom protokolu HTTP, pričom server vystavuje jednotlivé „endpointy“ vo forme REST API (z angl. Representational State Transfer Application Programming Interface). Tie slúžia na vykonávanie operácií ako prihlásenie používateľa, vytvorenie a uloženie portfólia, či získanie grafov a štatistík. Na výmenu dát medzi klientom a serverom sa používa formát JSON, ktorý je v prostredí webových aplikácií považovaný za štandard vďaka svojej čitateľnosti a jednoduchosti (Red Hat, 2020).

Takto navrhnutá architektúra umožňuje jednoduché pridávanie nových funkcií do aplikácie a zároveň zabezpečuje oddelenie zobrazovania údajov od ich spracovania na pozadí. Využitie RESTful API otvára možnosť prepojenia aplikácie s externými službami alebo s modernými frontendovými frameworkami, ako sú napríklad React či Vue.js.

3.8 Ekonomické ukazovatele

Aby sme vedeli posúdiť, ako si naše investície vedú, je dobré sledovať niekoľko základných ukazovateľov. Tie nám pomáhajú zistiť, či portfólio rastie, ako veľmi kolíše jeho hodnota a aké riziko vlastne podstupujeme. Medzi najčastejšie sledované patrí volatilita, ktorá hovorí o výkyvoch cien, YTD výnos (teda výnos od začiatku roka) a variancia, ktorá je úzko spätá s celkovým rizikom. V nasledujúcich častiach si tieto pojmy vysvetlíme jednoduchým spôsobom a ukážeme si, prečo sú dôležité pri hodnotení portfólia.

3.8.1 Volatilita v investičnom portfóliu

Volatilita predstavuje mieru, do akej sa cena finančného aktíva mení v čase. V oblasti investovania sa často používa ako indikátor rizika, pretože vyššia volatilita môže naznačovať väčšiu neistotu na trhu. Je dôležité si uvedomiť, že volatilita neodzrkadľuje smer pohybu cien, ale skôr rozsah týchto pohybov. To znamená, že aktívum s vysokou volatilitou môže zaznamenať výrazné cenové výkyvy v oboch smeroch – nahor aj nadol. Vysoká volatilita často súvisí s obdobím zvýšenej neistoty alebo strachu medzi investormi. Napríklad index VIX, známy ako „index strachu“, meria očakávanú volatilitu trhu na základe cien opcií na index S&P 500. V období poklesu akciového trhu má VIX tendenciu stúpať, čo naznačuje zvýšenú volatilitu a obavy investorov (HAYES Adam, 2024).

V kontexte portfóliového manažmentu sa volatilita najčastejšie počíta ako štandardná odchýlka denných výnosov za zvolené časové obdobie, ktorá sa následne anualizuje vynásobením odmocninou z počtu obchodných dní v roku (bežne sa používa hodnota 252). Vzorec na výpočet ročnej volatility:

$$\sigma_{ročná} = \sigma_{denné\ výnosy} \times \sqrt{T} \tag{3.1}$$

Kde: $\sigma_{ročná}$ je ročná volatilita, $\sigma_{denné\ výnosy}$ je štandardná odchýlka denných výnosov, T je počet obchodných dní v roku. Vo finančných aplikáciách, ako je táto, sa volatilita vypočítava z historických údajov o zatváracích cenách akcií. Výsledná štandardná odchýlka týchto denných výnosov poskytuje mieru rozptylu, ktorá sa následne škáluje na ročnú bázu.

3.8.2 YTD (Year-To-Date) výnos

YTD výnos predstavuje zmenu hodnoty finančného aktíva od začiatku kalendárneho roka po aktuálny dátum. Je to často používaný ukazovateľ na rýchle zhodnotenie výkonnosti investície v aktuálnom roku. YTD výnos sa zvyčajne vyjadruje v percentách a pomáha investorom rýchlo zistiť, či ich investícia rastie, stagnuje, alebo klesá (Daniel Liberto, 2025).

Sledovanie YTD výnosu umožňuje investorom rýchlo identifikovať, ktoré aktíva vykazujú najlepšiu výkonnosť v aktuálnom roku, čo je veľmi užitočné pri porovnávaní rôznych investícií. Tento ukazovateľ je tiež často používaný pri výpočtoch a analýzach výkonnosti portfólií, pretože poskytuje jednoduchý spôsob, ako posúdiť, či portfólio dosahuje požadované výnosy na ročnej báze. YTD výnos sa vypočíta ako percentuálna zmena medzi počiatočnou cenou (na začiatku roka) a aktuálnou cenou daného aktíva podľa vzorca:

$$YTD = \frac{P_{current} - P_{start}}{P_{start}} \times 100 \tag{3.2}$$

Kde: YTD sú výnosy od začiatku roka, $P_{current}$ je aktuálna cena cenného papiera alebo aktíva k dnešnému dňu a P_{start} je počiatočná cena cenného papiera alebo aktíva na začiatku aktuálneho roka.

Pri investičnej aplikácii, ako je táto, je YTD výnos dôležitou súčasťou vizualizácie výkonnosti jednotlivých akcií v portfóliu. Používateľ tak môže ľahko porovnať relatívne zhodnotenie aktív a zistiť, ktoré z nich prinášajú najvyšší výnos v aktuálnom roku. Tento ukazovateľ sa dynamicky prepočítava z dát načítaných prostredníctvom knižnice yFinance.

3.8.3 Variancia a riziko portfólia

Pri budovaní portfólia je dôležité sledovať nielen výnosy, ale aj riziko. Variancia portfólia meria mieru rozptylu kombinovaných výnosov jednotlivých aktív. Na jej výpočet sa používa nielen volatilita jednotlivých aktív, ale aj korelácie medzi nimi. Táto metóda je kľúčová pre diverzifikáciu. Cieľom je vytvoriť portfólio, ktoré má nižšie kolísanie pri zachovaní očakávaného výnosu (HAYES Adam, 2024).

Vo webovej aplikácii sa výpočet rizika realizuje pomocou zjednotenia denných výnosov každého aktíva a ich následného váženia podľa pridelených alokácií. Výsledkom je celkové riziko portfólia, ktoré používateľ vidí spolu s ďalšími výkonnostnými metrikami. Táto funkcionalita prispieva k lepšiemu pochopeniu diverzifikácie a finančných súvislostí pri správe investícií. Vzorec pre výpočet variancie portfólia:

$$Var(R_p) = \sum_{i=1}^{n} w_i^2 \cdot Var(R_i) + \sum_{i \neq j} w_i w_j \cdot Cov(R_i, R_j) \quad (3.3)$$

Kde: $Var(R_p)$ je variancia portfólia, w_i a w_j sú váhy jednotlivých aktív v portfóliu, $Var(R_i)$ je variancia jednotlivého aktíva, $Cov(R_i, R_j)$ je kovariancia medzi dvoma aktívami i a j.

3.9 Význam sektorov v investovaní

Aby človek dokázal zodpovedne investovať, mal by rozumieť tomu, ako sa finančný trh delí podľa sektorov. Každý sektor predstavuje určitú oblasť hospodárstva, ktorá má svoje vlastné špecifiká. Niektoré sektory sú stabilné a odolné voči výkyvom, iné sú zas dynamickejšie, no rizikovejšie. Práve tieto rozdiely spôsobujú, že jednotlivé sektory reagujú na ekonomické zmeny rôzne. Vďaka tomu je možné investície rozumne rozložiť, a tým znížiť celkové riziko portfólia.

Každý sektor, ktorý sa na finančných trhoch obchoduje, má svoje špecifiká a funguje trochu inak. Napríklad energetika patrí medzi stabilnejšie oblasti. Firmy, ktoré vyrábajú a distribuujú elektrinu, plyn či ropu, sú dôležité v každom období, takže ich výkonnosť nebýva príliš závislá od výkyvov v ekonomike. Podobne je na tom sektor základných spotrebných potrieb. Sem patria napríklad výrobcovia potravín alebo hygienických výrobkov. Ľudia tieto produkty potrebujú neustále, bez ohľadu na to, ako sa darí hospodárstvu. Naopak, priemyselný sektor už závisí viac od ekonomického cyklu. Keď sa darí ekonomike, rastie dopyt po stavebných strojoch, dopravných prostriedkoch či ďalších priemyselných výrobkoch. Ešte citlivejší na vývoj trhu je sektor spotrebiteľského tovaru, kam patrí napríklad móda, elektronika alebo automobily. Ak prídu horšie časy, ľudia tieto nákupy často odkladajú. Finančný sektor, teda banky a poisťovne, reaguje najmä na úrokové sadzby a zásahy regulátorov. Nehnuteľnosti sú citlivé najmä na vývoj úrokov – keď sú sadzby vysoké, hypotéky sú drahšie a dopyt po bývaní klesá. Technologický sektor je zas rýchlo rastúci, inovatívny a často láka investorov, ktorí hľadajú vyššie výnosy. Zdravotníctvo je naopak jednou z najstabilnejších oblastí – ľudia potrebujú zdravotnú starostlivosť bez ohľadu na stav ekonomiky. Komunikačné služby reagujú najmä na nové technológie a zmeny v správaní spotrebiteľov.

Ak chce investor vytvoriť vyvážené portfólio, mal by myslieť na zastúpenie rôznych sektorov. Práve to je základ diverzifikácie – ak sa jednému odvetviu prestane dariť, iné ho môžu „podržať“. Stabilnejšie sektory, ako napríklad energetika alebo zdravotná starostlivosť, dokážu portfólio ochrániť v horších časoch. Naopak, technologický sektor síce prináša vyššiu mieru rizika, ale ponúka aj výrazne väčší potenciál zhodnotenia.

4 Výsledky práce

V tejto časti práce sa podrobne zameriame na jednotlivé fázy vývoja aplikácie, pričom postupne rozoberieme technické riešenia, ktoré boli pri jej implementácii použité. Cieľom tejto kapitoly je detailne vysvetliť kľúčové časti zdrojového kódu, zdôvodniť výber konkrétnych technológií a objasniť rozhodnutia prijaté počas vývoja webstránky.

4.1 Inštalácia Flasku a základných knižníc

Základom serverovej časti aplikácie je mikroframework Flask. Ako hlavné vývojové prostredie sme si zvolili Visual Studio Code, ktoré v kombinácii s dostupnými rozšíreniami pre Python a Flask výrazne zjednodušilo a zrýchlilo samotnú implementáciu. Toto prostredie nám zároveň umožnilo pohodlnú prácu so zdrojovým kódom, verzovanie, ako aj spúšťanie a ladenie aplikácie počas vývoja.

Najskôr sme nainštalovali framework Flask a k nemu prislúchajúce rozšírenia. Inštalácia knižníc prebiehala prostredníctvom príkazového riadku s využitím balíčkového inštalatéra pip, ktorý je štandardnou súčasťou prostredia Python. Prvým krokom bolo nainštalovať knižnice potrebné na samotné spustenie a správu aplikácie: *pip install flask flask-login flask- wtf flask-bcrypt flask-sqlalchemy flask-migrate*

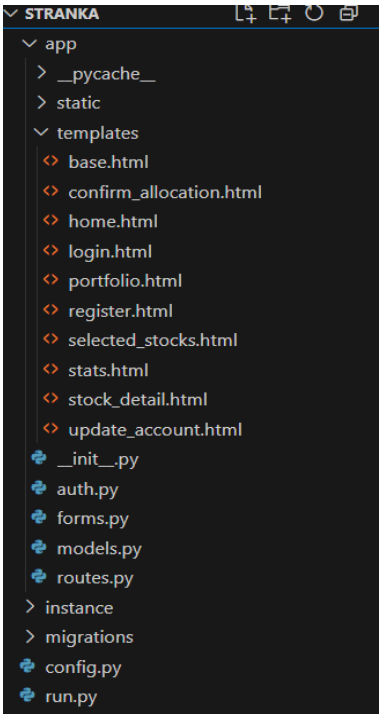
Tieto knižnice predstavujú základnú infraštruktúru aplikácie. Konkrétne Flask zabezpečuje samotný webový server a routovanie požiadaviek, Flask-Login slúži na správu session a autentifikáciu používateľov, Flask-WTF uľahčuje prácu s formulármi a zabezpečuje ich validáciu, Flask-Bcrypt zabezpečuje bezpečné šifrovanie hesiel, Flask-SQLAlchemy umožňuje jednoduchú prácu s databázou pomocou ORM a Flask-Migrate slúži na správu migrácií pri zmenách štruktúry databázy. Tieto komponenty tvoria robustný základ pre vývoj webovej aplikácie, ktorý zabezpečuje jej správne fungovanie, bezpečnosť a efektívnu správu dát.

Neskôr boli do projektu doplnené knižnice, ktoré boli kľúčové pre spracovanie finančných údajov, vizualizáciu výstupov a načítavanie dát zo súborov typu Excel. Ich inštalácia sa realizovala prostredníctvom príkazového riadku príkazom: *pip install plotly yfinance pandas flask-paginate*. Z hľadiska funkcionality Plotly umožňuje vizualizáciu údajov a interakciu používateľa s aplikáciou prostredníctvom vytvárania interaktívnych

grafov, ako sú koláčové grafy sektorového rozdelenia alebo vývoj portfólia, pričom YFinance slúži na získavanie aktuálnych, ako aj historických dát o akciách a ETF z Yahoo Finance, čo je základ pre výpočty volatility, YTD výnosov a výkonnosti daného portfólia. Pandas taktiež poskytuje nástroje pre analýzu dát a ich transformáciu. Stránkovanie záznamov pri dlhších výpisoch zabezpečuje Flask-paginate, a tak umožňuje efektívne zobrazenie a navigáciu cez väčšie množstvo dát.

Po úspešnej inštalácii knižníc sme vytvorili základnú štruktúru projektu. Visual Studio Code nám v tomto smere poskytol dobrú prehľadnosť a možnosť pohodlne pracovať so všetkými priečinkami a súbormi. Štruktúra projektu bola navrhnutá a členená tak, ako je znázornené na obrázku 2.

Obrázok 2: Štruktúra projektu vo Visual Studio Code



Zdroj: Vlastné spracovanie

Priečinok app obsahuje všetky základné časti aplikácie ako routy, databázové modely, formuláre, zložky pre HTML šablóny a statické súbory. Po nainštalovaní potrebných balíčkov a vytvorení štruktúry projektu bola centrálna časť aplikácie sústredená do súboru *__init__.py*, ktorý sa nachádza v hlavnom priečinku app/. Tento súbor slúži ako vstupný bod na vytvorenie a nakonfigurovanie samotnej Flask aplikácie pomocou funkcie *create_app()*.

V tejto časti sa vytvára hlavná aplikácia pomocou Flasku a zároveň sa k nej pripájajú všetky potrebné veci, ktoré aplikácia potrebuje na fungovanie – teda databáza, prihlasovanie používateľov či správa zmien v databázovej štruktúre. Táto funkcia zabezpečí, že všetko spolu správne komunikuje a aplikácia je pripravená na spustenie.

Inicializujú sa rozšírenia: *SQLAlchemy, Flask-Migrate, Flask-Bcrypt a Flask-Login*. Keď sú všetky rozšírenia pripravené, aplikácia si načíta tzv. blueprint s názvom routes. V ňom sú uložené všetky dôležité odkazy - teda URL adresy a logika, ktorá sa za nimi skrýva. Tu sa rozhoduje, čo sa má zobraziť, keď používateľ klikne na konkrétnu stránku. Konfiguráciu a inicializáciu rozšírení je možné vidieť na obrázku 3.

Obrázok 3: __init__.py – konfigurácia aplikácie

__init__.py > ...

```python
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from flask_login import LoginManager
from flask_migrate import Migrate

# Inicializácia rozšírení
db = SQLAlchemy()
bcrypt = Bcrypt()
login_manager = LoginManager()
migrate = Migrate()

login_manager.login_view = 'routes.login'


def create_app():
    app = Flask(__name__)
    app.config['SECRET_KEY'] = 'kjill447'
    app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'

    # Inicializácia databázy a migrácií
    db.init_app(app)
    migrate.init_app(app, db)
    bcrypt.init_app(app)
    login_manager.init_app(app)

    from app import routes
    app.register_blueprint(routes.bp)

    return app
```

Zdroj: Vlastné spracovanie

Aplikácia sa následne spúšťala príkazom: *python run.py*. Tento krok otvoril aplikáciu na adrese *http://127.0.0.1:5000/*, kde sme ju mohli počas celého vývoja testovať a priebežne dopĺňať o nové funkcionality. Tento spôsob práce umožnil rýchle ladenie, prehľadnosť a flexibilitu pri vývoji backendovej časti aplikácie.

4.2 Správa používateľských účtov

Správa používateľských účtov patrí medzi základné súčasti každej aplikácie, pretože vďaka nej môžu všetci používatelia bezpečne pracovať so svojimi vlastnými dátami. Možnosť registrácie, prihlásenia sa, správa používateľskej relácie, ako aj ochrana prihlasovacích údajov sú zahrnuté v danej funkcionalite. Používateľ má po prihlásení prístup ku vlastnému investičnému portfóliu ako aj štatistikám predstavujúcim napríklad zloženie portfólia, výnos alebo volatilita.

4.2.1 Registrácia a prihlasovanie

Na správu prihlasovania a registrácie sme využili rozšírenia Flask-Login a Flask-WTF, ktoré umožňujú jednoduchú tvorbu a validáciu formulárov. Heslá sú pri registrácii šifrované pomocou Flask-Bcrypt, čo zvyšuje bezpečnosť uložených údajov.

Pri registrácii používateľ vyplní používateľské meno, e-mail a heslo, ktoré sa po úspešnej validácii uložia do databázy. Na overenie údajov slúžia aj vlastné validačné metódy, ktoré kontrolujú jedinečnosť mena a e-mailu. Pri prihlasovaní sa overí, či existuje záznam so zadaným používateľským menom a či zadané heslo zodpovedá zašifrovanému heslu v databáze.

Na vytváranie a spracovanie formulárov slúži modul Flask-WTF. V projekte boli vytvorené tri základné triedy formulárov:

- RegistrationForm – pre registráciu používateľa,
- LoginForm – pre prihlásenie používateľa,
- UpdateAccountForm – pre aktualizáciu používateľských údajov.

Každý formulár obsahuje preddefinované validátory, ktoré overujú požadované vlastnosti vstupov. Konkrétne pri registrácii je to e-mailová adresa, dĺžka mena, zhodnosť hesiel. Na obrázku 4 je definícia týchto formulárov uvedená v kóde aplikácie.

Obrázok 4: Definícia formulárov pre registráciu, prihlásenie a aktualizáciu účtu

```python
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import DataRequired, Email, EqualTo, Length

class RegistrationForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired(), Length(min=2, max=20)])
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    confirm_password = PasswordField('Confirm Password', validators=[DataRequired(), EqualTo('password')])
    submit = SubmitField('Sign Up')

class LoginForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    submit = SubmitField('Login')


class UpdateAccountForm(FlaskForm):
    username = StringField("Username", validators=[DataRequired(), Length(min=2, max=20)])
    password = PasswordField("New Password", validators=[Length(min=3)])
    confirm_password = PasswordField("Confirm Password", validators=[EqualTo("password")])
    submit = SubmitField("Update Account")
```

Zdroj: Vlastné spracovanie

Všetky hlavné funkcie ako registrácia, prihlásenie a aktualizácia účtu sú implementované v súbore routes.py. Registrácia používateľa sa vykonáva prostredníctvom route *@bp.route("/register"),* kde systém najprv overí, či užívateľ so zadanou e-mailovou adresou existuje. Ak nie, vytvorí sa nový účet so zašifrovaným heslom. Po úspešnej registrácii je používateľ presmerovaný na prihlasovaciu stránku ako je znázornené na obrázku 5.

Obrázok 5: Registrácia používateľa a vytvorenie účtu v routes.py

```python
@bp.route("/register", methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        existing_user = User.query.filter_by(email=form.email.data).first()
        if existing_user:
            flash('An account with this email already exists.', 'danger')
            return redirect(url_for('routes.register'))

        hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user = User(username=form.username.data, email=form.email.data, password=hashed_password)
        db.session.add(user)
        db.session.commit()
        flash('Your account has been created!', 'success')
        return redirect(url_for('routes.login'))

    return render_template('register.html', title='Register', form=form)
```

Zdroj: Vlastné spracovanie

Route *@bp.route("/login")* slúži na overenie prihlasovacích údajov a prihlásenie. Ak zadaný e-mail existuje v databáze a heslo je správne (porovnané cez bcrypt), používateľ je autentifikovaný pomocou funkcie *login_user()* a následne presmerovaný na domovskú stránku. V opačnom prípade sa zobrazí chybová hláška , čo môžeme vidieť na obrázku 6.

Obrázok 6: Prihlasovanie používateľa a autentifikácia pomocou Flask-Login

```python
@bp.route("/login", methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('routes.home'))

    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user and bcrypt.check_password_hash(user.password, form.password.data):
            login_user(user)
            flash('You have been logged in!', 'success')
            return redirect(url_for('routes.home'))  # presmerovanie po prihlásení
        else:
            flash('Login Unsuccessful. Please check email and password')
    return render_template('login.html', title='Login', form=form)
```

Zdroj: Vlastné spracovanie

Na obrázku 7 je znázornený spôsob, akým aplikácia spracováva požiadavky na zmenu prihlasovacích údajov používateľa.

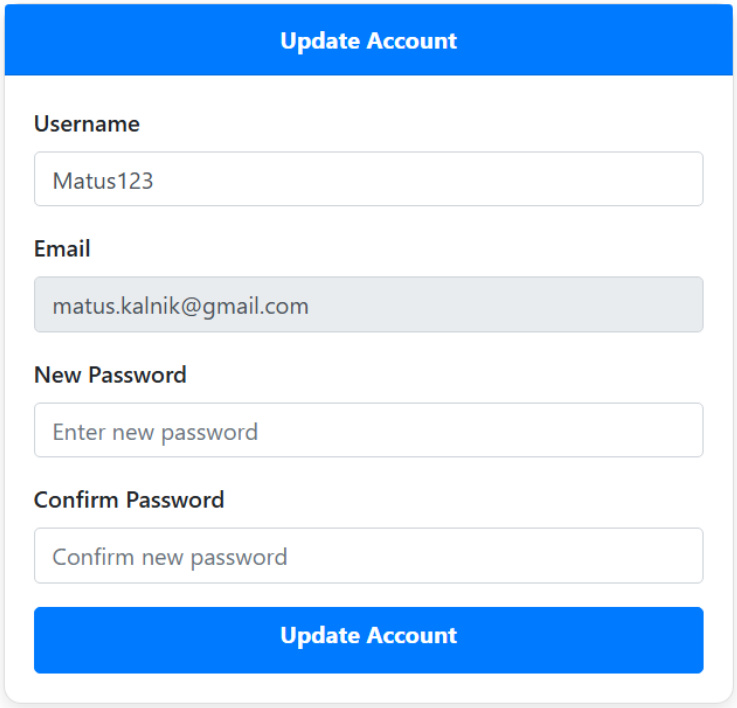
Obrázok 7: Aktualizácia používateľského účtu v routes.py – zmena údajov

```python
@bp.route("/update_account", methods=["GET", "POST"])
@login_required
def update_account():
    form = UpdateAccountForm()
    if form.validate_on_submit():
        if form.username.data:
            current_user.username = form.username.data
        if form.password.data:
            hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
            current_user.password = hashed_password
        db.session.commit()
        flash("Your account has been updated!", "success")
        return redirect(url_for("routes.update_account"))
    elif request.method == "GET":
        form.username.data = current_user.username

    return render_template("update_account.html", title="Update Account", form=form)
```

Zdroj: Vlastné spracovanie

Na obrázku 8 je formulár *UpdateAccountForm*, ktorý umožňuje aktualizáciu používateľského mena a hesla, pričom sa najprv overí platnosť zadaných údajov pomocou preddefinovaných validátorov. Ak používateľ zadá nové meno alebo heslo a údaje prejdú validáciou, dôjde k ich aktualizácii v databáze. Heslo je pred uložením bezpečne zašifrované pomocou knižnice *bcrypt*. Pri načítaní stránky metódou GET sa formulár automaticky predvyplní aktuálnym používateľským menom. Celá route je chránená dekorátorom *@login_required*, ktorý zabezpečuje, že prístup na túto stránku majú výlučne prihlásení používatelia.

Obrázok 8: Formulár na zmenu prihlasovacích údajov



Zdroj: Vlastné spracovanie

4.2.2 *Uchovávanie používateľských relácií a zabezpečenie prístupu*

Na správu prihlásenia používateľa a uchovávanie relácií počas aktívnej session sme využili rozšírenie *Flask-Login*, ktoré sme už spomínali v súvislosti s prihlasovaním. Toto rozšírenie automaticky zabezpečuje, že prihlásený používateľ zostáva overený počas celej relácie prostredníctvom cookies. Prístup k chráneným častiam aplikácie (napr. správa portfólia či zobrazenie štatistík) je riešený pomocou dekorátora *@login_required*. Vďaka tomu je neprihlásený používateľ automaticky presmerovaný na prihlasovaciu stránku. Login manager je nakonfigurovaný v súbore *__init__.py* na obrázku 3.

4.2.3 *Odhlásenie a ukončenie relácie*

Aby mal používateľ možnosť bezpečne ukončiť svoju reláciu, do aplikácie sme pridali funkciu odhlásenia. Slúži na to príkaz *logout_user()*, ktorý rovnako

ako prihlásenie poskytuje rozšírenie Flask-Login, ktoré už bolo spomenuté v predchádzajúcich častiach.

Obrázok 9: Implementácia odhlásenia používateľa v súbore routes.py

```
@bp.route("/logout")
def logout():
    logout_user()
    return redirect(url_for('routes.login'))
```

Zdroj: Vlastné spracovanie

4.3 Získavanie údajov a príprava databázy

Predtým, než sa používateľ pustí do zostavovania vlastného investičného portfólia, bolo potrebné pripraviť databázu, z ktorej si bude môcť vyberať. Táto databáza obsahuje základné informácie o jednotlivých akciách. Každý titul je v nej označený tickerom, čo je krátky kód väčšinou skratka z veľkých písmen, ktorý ho identifikuje na burze. Príkladmi sú napríklad *AAPL* pre Apple alebo *MSFT* pre Microsoft. Kód je jednoduchý, ale podstatný pri načítavaní údajov z externých zdrojov. Názov (name), reprezentuje celé meno spoločnosti alebo fondu, čo umožňuje používateľovi jednoduchšie rozoznať, o aký subjekt sa jedná. Ďalšou položkou je sektor, ktorý určuje, do akej oblasti hospodárstva daná spoločnosť patrí. Tento atribút zohráva kľúčovú úlohu pri hodnotení diverzifikácie portfólia. Rozdelenie podľa sektorov a ich význam je detailnejšie popísaný v samostatnej teoretickej podkapitole 3.9. Posledným atribútom je krajina (country), ktorá označuje geografický pôvod spoločnosti. Táto informácia pomáha pri geografickej diverzifikácii portfólia a zohľadnení rizík spätých s konkrétnymi trhmi či ekonomikami.

Keďže knižnica yfinance nedokáže pri hromadnom načítaní vždy spoľahlivo poskytnúť údaje o sektore alebo krajine, bol manuálne vytvorený vlastný Excel súbor obsahujúci tieto informácie pre 64 vybraných akcií a ETF fondov (viď tabuľku 1). Tento súbor je základným zdrojom údajov pre aplikáciu a slúži na načítanie zoznamu titulov, ktoré sú následne dostupné používateľovi pri zostavovaní jeho investičného portfólia.

Tabuľka 1: Ukážka vstupnej databázy pre výber portfólia

Ticker	Name	Sector	Country
EMR	Emerson Electric	Industrials	USA
ETR	Entergy Corp.	Utilities	USA
EOG	EOG Resources	Energy	USA
EFX	Equifax Inc.	Financials	USA

Zdroj: vlastné spracovanie

Zoznam akcií a ETF bol manuálne vytvorený a overený na základe verejne dostupných informácií z portálov ako Yahoo Finance a oficiálnych webových stránok jednotlivých spoločností. Tento súbor bol následne zaradený do projektu a v ďalšej fáze spracovaný pomocou knižnice *pandas*, ako je znázornené na obrázku 10.

Obrázok 10: Načítanie Excel súboru s údajmi o akciách

```
import pandas as pd

FILE_PATH = 'C:/Users/Matúš Kalník/Desktop/stocks.xlsx'
```

Zdroj: Vlastné spracovanie

Zatiaľ čo sa údaje ako sektor a krajina načítavajú z pripraveného Excel súboru, aktuálne trhové ceny akcií sa získavajú priamo z internetu pomocou knižnice yfinance. Na tento účel bola vytvorená jednoduchá pomocná funkcia s názvom *fetch_stock_price*. Táto funkcia sa postará o to, aby pre každú vybranú akciu (ticker) stiahla jej aktuálnu zatváraciu cenu. V prípade, že sa údaje nepodarí získať, funkcia zachytí chybu a vráti hodnotu 'Error'.

Obrázok 11 : Funkcia fetch_stock_price – získanie aktuálnej ceny akcie z Yahoo Finance

```
def fetch_stock_price(ticker):
    try:
        stock = yf.Ticker(ticker)
        price = stock.history(period='1d')['Close'][0]
        return ticker, price
    except Exception as e:
        print(f"Error fetching data for {ticker}: {e}")
        return ticker, 'Error'
```

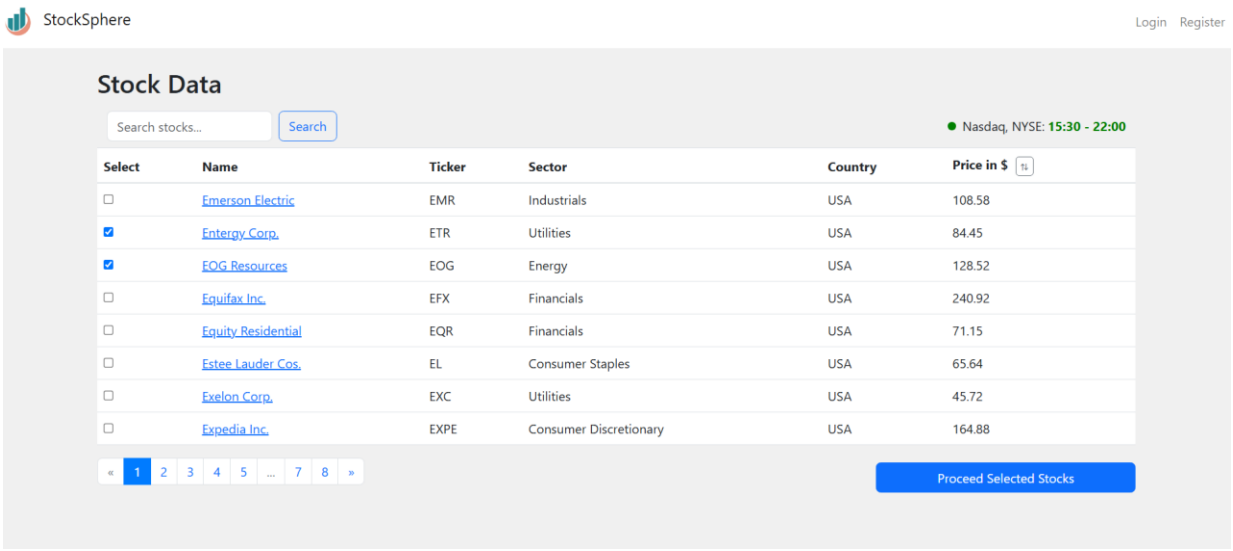
Zdroj: Vlastné spracovanie

4.4 Zobrazenie údajov na hlavnej stránke

Ako je znázornené na obrázku 12, na hlavnej stránke aplikácie sa nachádza prehľadná tabuľka, v ktorej sú zobrazené všetky dostupné akcie a ETF fondy načítané z databázy. Každý riadok tabuľky obsahuje základné informácie o titule – názov spoločnosti, ticker, sektor, krajinu pôvodu a aktuálnu cenu v dolároch, ktorá sa získava dynamicky z Yahoo Finance.

Údaje sa na hlavnú stránku aplikácie načítavajú z Excel súboru, Tento súbor sa načíta pomocou knižnice pandas, pričom sa z neho vytvorí dátový rámec (DataFrame). Následne sa z každej položky v tabuľke vytvorí záznam vo forme slovníka, ktorý sa pridáva do poľa *tickers_data*. Súbežne sa pre každý ticker zavolá funkcia *fetch_stock_price*, ktorá zabezpečí získanie aktuálnej trhovej ceny z Yahoo Finance. Výsledkom tohto procesu je pole *tickers_data*, ktoré obsahuje všetky informácie potrebné pre zobrazenie na hlavnej stránke.

Obrázok 12 : Zobrazenie údajov o akciách a ETF na hlavnej stránke aplikácie



Zdroj: Vlastné spracovanie

Používateľ má na hlavnej stránke možnosť vyhľadávať akcie podľa názvu spoločnosti alebo tickeru. Po zadaní textu do vyhľadávacieho poľa sa tento text porovnáva s každým záznamom v zozname. Hľadá sa výskyt reťazca (bez ohľadu na veľkosť písmen) buď v názve alebo v tickeri. Ak sa žiadny záznam nezhoduje s hľadaným výrazom, premenná no_results sa nastaví na True – čo môže byť neskôr použité na zobrazenie hlášky „Nenašli sa žiadne

výsledky“. Po filtrovaní nasleduje stránkovanie výsledkov. Počet záznamov na stránku je pevne nastavený na osem. Aktuálne číslo stránky je načítané z URL pomocou *request.args.get('page').* Na jeho základe sa vypočíta rozsah údajov, ktoré sa majú zobraziť na danej stránke – teda iba časť zoznamu akcií, ktorá zodpovedá aktuálnej stránke. Na vizuálne zobrazenie stránkovania bol použitý modul flask-paginate. Logika vyhľadávania a stránkovania záznamov je znázornená na obrázku 13.

Obrázok 13: Logika vyhľadávania a stránkovania záznamov

```
if search_query:
    tickers_data = [data for data in tickers_data if search_query in data['Name'].lower() or search_query in data['Ticker'].lower()]
    if not tickers_data:  # Ak nie sú žiadne výsledky
        no_results = True  # Nastavíme, že neboli nájdené výsledky

page = request.args.get('page', 1, type=int)
per_page = 8
total = len(tickers_data)
paginated_data = tickers_data[(page - 1) * per_page: page * per_page]

pagination = Pagination(page=page, per_page=per_page, total=total, record_name='stocks', css_framework='bootstrap4')
```

Zdroj: Vlastné spracovanie

Na hlavnej stránke aplikácie je integrovaný prvok, ktorý informuje používateľa o aktuálnom stave americkej burzy (NASDAQ a NYSE) viď obrázok 12. Ide o vizuálny indikátor, ktorý zobrazuje, či je burza práve otvorená, alebo zatvorená, a zároveň uvádza presné otváracie hodiny v stredoeurópskom čase (15:30 – 22:00).

Táto funkcionalita je zabezpečená pomocou JavaScript skriptu, ktorý každú sekundu zisťuje aktuálny čas podľa časovej zóny „Europe/Vienna“. Na základe toho, aký je deň v týždni a koľko je hodín, vyhodnotí, či sa nachádzame v rámci obchodných hodín amerických búrz. Skript zároveň porovnáva aktuálny dátum so zoznamom sviatkov (napr. Nový rok, Deň nezávislosti, Vianoce a pod.), počas ktorých je obchodovanie pozastavené.

Ak je burza otvorená, text aj vizuálny indikátor (malá kruhová ikona) sa zafarbia na zeleno. Naopak, ak je zatvorená, zobrazí sa červená farba. Tento jednoduchý, no praktický prvok zabezpečuje, aby používateľ vždy vedel, či sú zobrazené ceny aktuálne pohyblivé, alebo ide o posledné zatváracie hodnoty z predchádzajúceho dňa. Skript sa vykonáva v pravidelných intervaloch pomocou funkcie *setInterval(),* vďaka čomu sa stav burzy aktualizuje automaticky a bez potreby obnovy stránky ako môžeme vidieť na obrázku 14.

Obrázok 14: Skript na kontrolu stavu americkej burzy v reálnom čase pomocou časovej zóny

```
<script>
  function updateMarketStatus() {
    const now = new Date();
    const viennaTime = new Intl.DateTimeFormat("de-AT", {
      timeZone: "Europe/Vienna",
      hour: "2-digit",
      minute: "2-digit",
      hour12: false
    }).format(now);

    const [hours, minutes] = viennaTime.split(":").map(Number);
    const day = now.getDay();
    const dateStr = now.toISOString().split('T')[0];

    const holidaysNasdaq = [
      "2025-01-01", "2025-01-20", "2025-02-17", "2025-04-18",
      "2025-05-26", "2025-06-19", "2025-07-04", "2025-09-01",
      "2025-11-27", "2025-12-25", "2025-12-31"
    ];

    const isHolidayNasdaq = holidaysNasdaq.includes(dateStr);
    const isOpenNasdaq = !isHolidayNasdaq && (day >= 1 && day <= 5) &&
      ((hours > 15 || (hours === 15 && minutes >= 30)) && (hours < 22));
```

Zdroj: Vlastné spracovanie

4.5 Tvorba investičného portfólia

Jednou z najdôležitejších funkcií aplikácie je zostavenie vlastného investičného portfólia. Používateľ si môže vybrať z databázy svetových akcií a ETF fondov, priradiť im vlastné váhové zastúpenie a celý výber si uložiť do databázy. Takto si môže jednoducho vytvoriť vlastnú investičnú stratégiu a následne sledovať, ako by sa jej darilo na základe aktuálnych aj historických údajov z finančných trhov.

4.5.1 Výber akcií pre tvorbu portfólia

Po načítaní a zobrazení zoznamu dostupných akcií a ETF má používateľ možnosť interaktívne si vybrať investičné tituly, ktoré chce zaradiť do svojho portfólia. Zaškrtávacie políčka sú umiestnené v ľavom stĺpci tabuľky a slúžia na výber požadovaných titulov. Tieto políčka sú prepojené s mechanizmom, ktorý uchováva výber aj pri prechádzaní medzi rôznymi stránkami tabuľky alebo pri vykonávaní vyhľadávania.

Pre lepšiu prehľadnosť a zjednodušenie rozhodovania umožňuje tabuľka zoradiť akcie podľa aktuálnej trhovej ceny. Po kliknutí na tlačidlo „Sort by Price“ sa riadky zoradia podľa ceny buď vzostupne, alebo zostupne.

Obrázok 15: Zoradenie riadkov tabuľky podľa ceny akcií pomocou JavaScriptu

```
//  Zoradenie akcií podľa ceny
sortBtn.addEventListener("click", function () {
    let rows = Array.from(stockTableBody.querySelectorAll("tr"));
    rows.sort((a, b) => {
        let priceA = parseFloat(a.cells[5].dataset.price) || 0;
        let priceB = parseFloat(b.cells[5].dataset.price) || 0;
        return ascending ? priceA - priceB : priceB - priceA;
    });

    ascending = !ascending; // Otočíme smer zoradenia
    rows.forEach(row => stockTableBody.appendChild(row)); // Aktualizujeme tabuľku
});
```

Zdroj: Vlastné spracovanie

Pri kliknutí na tlačidlo z obrázku 12 „Proceed Selected Stocks“ sa všetky vybrané akcie automaticky načítajú a pridajú do tela formulára ako skryté vstupy čo môžeme vidieť na obrázku 16. Tento mechanizmus zabezpečuje, že po odoslaní formulára na server sa všetky označené tituly správne prenesú a ďalej spracujú v rámci tvorby portfólia. Po prihlásení sa výber viaže na konkrétneho používateľa. V prípade, že sa používateľ odhlási alebo zmení účet, sessionStorage sa vymaže.

Obrázok 16: Pridanie vybraných akcií do formulára pri kliknutí na „Proceed Selected Stocks“

```
// ✅ Pred odoslaním formulára pridá všetky vybrané akcie
document.getElementById("stocksForm").addEventListener("submit", function () {
    let selectedStocks = new Set(JSON.parse(sessionStorage.getItem("selectedStocks")) || []);
    selectedStocks.forEach(ticker => {
        let hiddenInput = document.createElement("input");
        hiddenInput.type = "hidden";
        hiddenInput.name = "selected_stocks";
        hiddenInput.value = ticker;
        this.appendChild(hiddenInput);
    });
});
```

Zdroj: Vlastné spracovanie

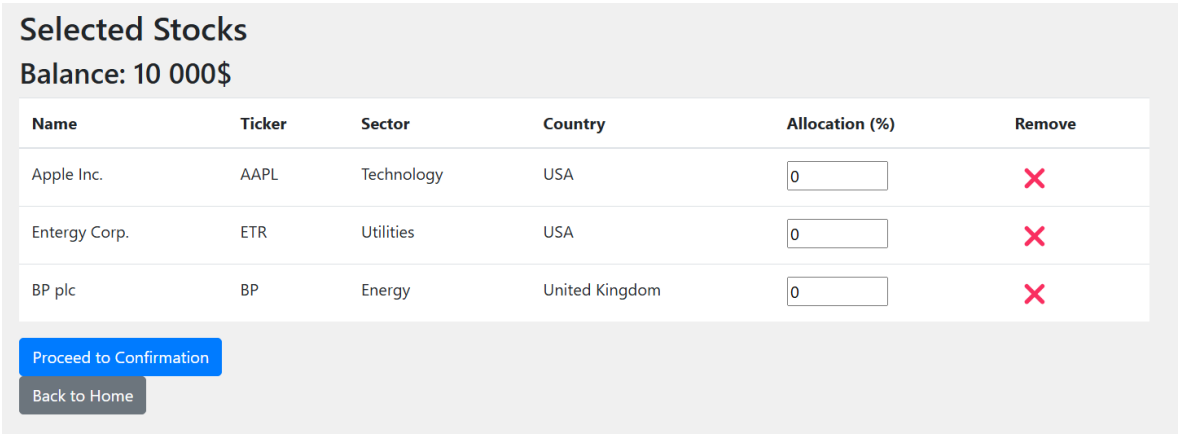
4.5.2 Pridelenie alokácie

V ďalšej fáze si používateľ určí, aké percentuálne zastúpenie bude mať každá z vybraných akcií alebo ETF fondov v jeho portfóliu. Táto alokácia vyjadruje rozdelenie celkového rozpočtu – v tejto aplikácii fixne stanoveného na 10 000 EUR – medzi jednotlivé tituly. Napríklad, ak používateľ pridelí jednej akcii 40 % a inej 60 %, do prvej investuje 4000 EUR a do druhej 6000 EUR.

Pre každú akciu sa zobrazí textové pole, do ktorého je možné zadať požadovanú hodnotu alokácie v percentách. Aplikácia je nastavená tak, aby v reálnom čase kontrolovala, či celkový súčet všetkých zadaných alokácií presne zodpovedá 100 %. V prípade, že používateľ zadá vyšší alebo nižší súčet, systém zobrazí validačné upozornenie a znemožní pokračovať v procese ukladania portfólia, kým nebude vstup opravený.

Na obrázku 17 je formulár pre priradenie percenta alokácie. V rámci návrhu rozhrania bolo dôležité, aby formulár zostal prehľadný. Jednotlivé polia sú zoradené podľa výberu a doplnené o názvy akcií a ich tickery, aby bolo jasné, že každá alokácia je správne priradená k príslušnému cennému papieru. Okrem toho sú automaticky kontrolované aj nesprávne znaky (napr. písmená) alebo záporné čísla, ktoré by narušili výpočet portfólia.

Obrázok 17: Formulár pre zadanie alokácie a úpravu vybraných akcií



Zdroj: Vlastné spracovanie

Okrem zadania alokácie má používateľ možnosť odstrániť akcie zo zoznamu výberu. V pravej časti tabuľky sa pri každej akcii nachádza červený krížik, ktorý slúži práve pre túto operáciu. Po kliknutí na daný symbol sa vykoná POST požiadavka na server s identifikátorom (tickerom) konkrétnej akcie, ktorá sa má odstrániť. Na strane servera sa následne aktualizuje obsah premenných, čím sa zabezpečí, že odstránená akcia okamžite zmizne z tabuľky aj z ďalšieho spracovania. Používateľ je o tejto akcii zároveň informovaný pomocou hlášky o úspešnom odstránení. Implementácia kódu je na obrázku 18.

Obrázok 18: Backendová logika pre odstránenie akcie z výberu pomocou session

```python
@bp.route("/selected_stocks_action", methods=["POST"])
def selected_stocks_action():
    selected_stocks = session.get("selected_data", [])

    ticker_to_remove = request.form.get("remove")
    if ticker_to_remove:
        if "selected_stocks" in session:
            session["selected_stocks"] = [t for t in session["selected_stocks"] if t != ticker_to_remove]
        if "selected_data" in session:
            session["selected_data"] = [st for st in selected_stocks if st["Ticker"] != ticker_to_remove]

        flash(f"Removed {ticker_to_remove} from selection.", "success")
        return redirect(url_for("routes.display_selected_stocks"))
```

Zdroj: Vlastné spracovanie

4.5.3 *Výpočet investovaných súm*

Po zadaní alokácie pre jednotlivé tituly systém automaticky vypočíta, aká presná suma z celkového investičného rozpočtu bude pridelená každému z nich. Na výpočet bol použitý nasledovný vzorec:

$$I = \frac{A}{100} \times B \tag{4.1}$$

Kde: I je investovaná suma, A je percentuálna alokácia daného aktíva, B je celkový rozpočet. Výpočty vychádzajú z fixne stanoveného rozpočtu, ktorý je v rámci tejto aplikácie nastavený na 10 000 EUR. Na obrázku 19 je zobrazené použitie vzorca v zdrojovom kóde.

Obrázok 19: Použitie vzorca pre výpočet investovaných súm v routes.py

```python
# Výpočet sumy investovanej do každej akcie
stock_investments = {ticker: (allocations[ticker] / 100) * total_budget for ticker in allocations}
```

Zdroj: Vlastné spracovanie

4.6 Potvrdenie portfólia a výpočet štatistík

Po zadaní alokácií používateľ prejde na stránku s potvrdením svojho portfólia. Tu získa podrobné údaje o investovaných sumách, výkonnosti portfólia a jednotlivých akcií. Systém zároveň vygeneruje grafy a analytické prepočty, ktoré pomáhajú lepšie porozumieť zvolenému investičnému rozloženiu. Táto časť zároveň slúži ako finálna rekapitulácia celého procesu tvorby portfólia – od výberu akcií, cez alokáciu, až po výpočet výnosov, volatility a

vizualizáciu vývoja investícií. Umožňuje používateľovi skontrolovať všetky vstupy a výstupy predtým, než dôjde k definitívnemu uloženiu portfólia do systému.

Obrázok 20: Zoznam investícií s výpočtom YTD výnosov a volatility

Confirm Allocation

Name	Ticker	Sector	Country	Allocation (%)	Investment ($)	YTD Return (%)	Volatility (%)
Equity Residential	EQR	Financials	USA	33.00%	3300.00 $	13.77%	0.27%
Estee Lauder Cos.	EL	Consumer Staples	USA	33.00%	3300.00 $	-61.72%	0.39%
Entergy Corp.	ETR	Utilities	USA	34.00%	3400.00 $	68.20%	0.24%

Portfolio Volatility

Metric	Value
Annual Portfolio Volatility	0.22%

Zdroj: Vlastné spracovanie

4.6.1 Výpočet YTD výnosov

YTD výnos (Year-To-Date) predstavuje zmenu ceny akcie od začiatku kalendárneho roka až po aktuálny deň. Tento ukazovateľ je veľmi obľúbený, keďže používateľovi umožňuje rýchlo zhodnotiť, ako sa akcia vyvíjala v priebehu aktuálneho roka.

Na obrázku 21 je výpočet YTD realizovaný nasledovne: pre každú akciu sa pomocou knižnice yfinance získajú historické dáta za posledný rok. Z týchto dát sa vezme cena na začiatku obdobia (prvý obchodný deň v roku) a cena na konci (najnovšia dostupná cena). Výnos sa vypočíta podľa vzorca 3.2. Ak pre niektorú akciu chýbajú historické údaje, výstup je označený ako „N/A“. Výsledky sa zobrazujú v tabuľke v percentách – kladné hodnoty sú zvýraznené zelenou, záporné červenou.

Obrázok 21: Použitie vzorca pre výpočet YTD výnos v routes.py

```
#  Výpočet YTD Return
if not stock_data_1y.empty:
    start_price = stock_data_1y.iloc[0]  # Cena na začiatku roka
    end_price = stock_data_1y.iloc[-1]  # Aktuálna cena
    ytd_return = ((end_price - start_price) / start_price) * 100
    ytd_returns[ticker] = round(ytd_return, 2)
else:
    ytd_returns[ticker] = "N/A"
```

Zdroj: Vlastné spracovanie

4.6.2 *Výpočet volatility akcií*

Volatilita je dôležitý ukazovateľ rizikovosti danej akcie – čím vyššia volatilita, tým viac sa cena v priebehu času mení. V aplikácii sa výpočet volatility vykonáva na základe historických cien akcie za obdobie 5 rokov. Najskôr sa vypočítajú denné výnosy pomocou percentuálnej zmeny cien *(funkcia .pct_change())*, a následne sa aplikuje štandardná odchýlka týchto výnosov. Pre prepočet na ročnú volatilitu sme si v kóde vytvorili konštantu *TRADING_DAYS,* kde je uložená hodnota 252, ktorá predstavuje počet obchodných dní v roku (približne 252). Výsledok sa následne vynásobí odmocninou z tejto konštanty a tento údaj sa zobrazí pre každú akciu v percentuálnom vyjadrení. Vzorec 3.1 je následne použitý, ako je zobrazené na obrázku 22.

Obrázok 22: Použitie vzorca pre výpočet volatility akcií v routes.py

```
#  Výpočet volatility (za posledných 5 rokov)
if not stock_data_5y.empty:
    daily_returns = stock_data_5y.pct_change().dropna()
    volatility[ticker] = round(daily_returns.std() * (TRADING_DAYS ** 0.5), 2)
else:
    volatility[ticker] = "N/A"
```

Zdroj: Vlastné spracovanie

4.6.3 *Výpočet volatility portfólia*

Volatilita celého portfólia sa odvodzuje podobne ako pri akciách, ale zohľadňuje sa aj alokácia jednotlivých titulov. Každý denný výnos je upravený podľa váhy, ktorú má daná akcia v portfóliu. Následne sa tieto vážené výnosy sčítajú a vypočíta sa ich štandardná odchýlka, ktorá je opäť prepočítaná na ročnú volatilitu. Vzorec pre jej výpočet je:

$$\sigma p = \sqrt{\sum_{i=1}^{n} w^2{}_i \cdot \sigma^2{}_i + \sum_{i=1}^{n} \sum_{j \neq i}^{n} w_i \cdot wj \cdot Cov(R_i, R_j)} \qquad (4.2)$$

Kde: σp je volatilita portfólia, w_i je váha i-tého aktíva v portfóliu, σ_i je volatilita i-tého aktíva, $Cov(R_i, R_j)$ je kovariancia medzi i-tým a j-tým aktívom, n je počet aktív v portfóliu.

4.6.4 Sektorová a geografická distribúcia

Na obrázku 23 je sektorové a geografické rozloženia portfólia v percentách. Aplikácia pracuje s údajmi zo súboru stocks.xlsx, ktorý ku každému titulu priraďuje informácie o sektore a krajine pôvodu. Po tom, ako si používateľ vyberie akcie a určí ich alokáciu, systém tieto alokácie automaticky agreguje podľa jednotlivých sektorov a krajín.

Inými slovami, aplikácia spočíta celkový percentuálny podiel každého sektora alebo krajiny na celkovom portfóliu na základe toho, ako boli alokované jednotlivé investície. Ak má napríklad používateľ v portfóliu tri rôzne akcie, z ktorých dve patria do technologického sektora s alokáciami 30 % a 20 %, a jedna patrí do sektora zdravotníctva s alokáciou 50 %, sektorová distribúcia bude 50 % technológie a 50 % zdravotníctvo.

Obrázok 23: Sektorová a geografická alokácia portfólia podľa priradených percentuálnych podielov

Sector Distribution

Sector	Allocation (%)
Health Care	50.00%
Information Technology	50.00%

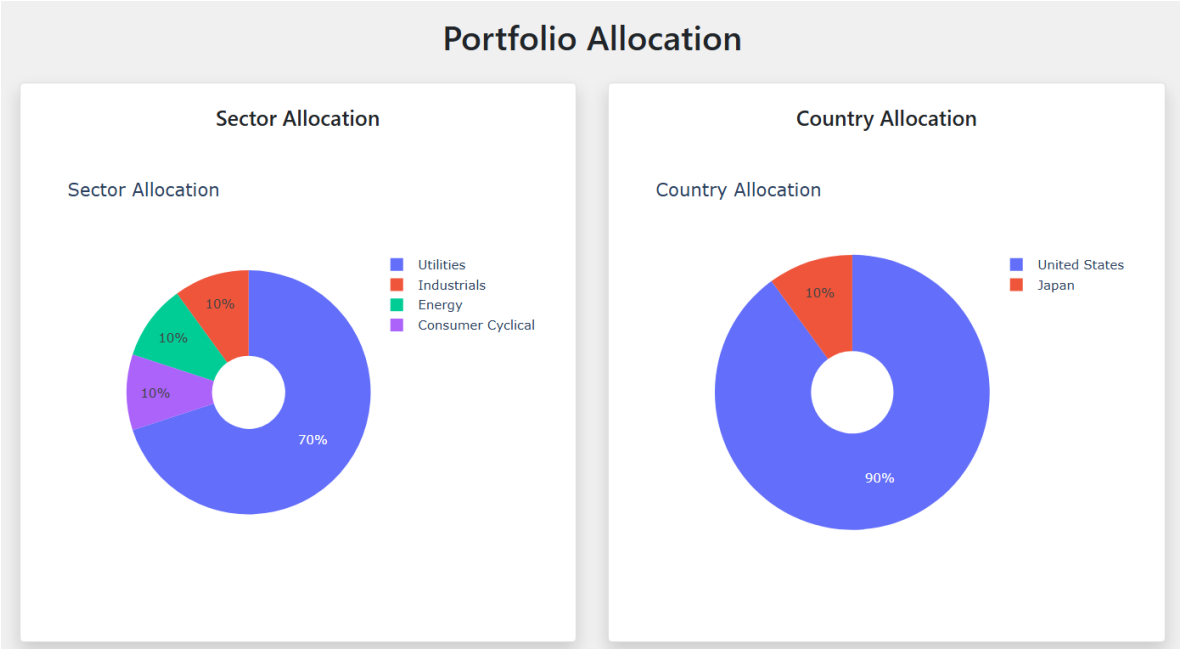
Country Distribution

Country	Allocation (%)
India	20.00%
Switzerland	50.00%
USA	30.00%

Zdroj: Vlastné spracovanie

Pre prihlásených používateľov je tento prehľad doplnený o grafické znázornenie vo forme koláčových grafov ako môžete vidieť na obrázku 24. Grafy sú prístupné v sekcii "Portfolio" po vytvorení portfólia a zobrazujú pomer alokácií podľa sektorov a krajín.

Obrázok 24: Grafické znázornenie sektorovej a krajinskej alokácie portfólia



Zdroj: Vlastné spracovanie

4.7 Grafy vývoja portfólia a akcií

V spodnej časti stránky potvrdenia alokácie z obrázku 20 sú používateľovi sprístupnené tri samostatné grafy, ktoré zobrazujú výkonnosť jeho investičného portfólia za posledných 5 rokov. Tieto grafy sú generované pomocou knižnice Plotly, ktorá umožňuje interaktívne zobrazenie dát priamo v prehliadači – vrátane možnosti priblíženia, zobrazenia hodnôt, zapínania/vypínania jednotlivých titulov a exportu do obrázkového formátu.

Na to, aby mohla aplikácia zobraziť vývoj celého portfólia v čase, je potrebné spočítať, ako by sa hodnota portfólia vyvíjala na základe historických cien jednotlivých titulov. K tomuto účelu sa využíva knižnica yfinance, ktorá umožňuje sťahovať historické údaje o zatváracích cenách akcií za posledných päť rokov. Pre každý vybraný titul sú tieto

dáta upravené podľa toho, akú alokáciu používateľ danému titulu priradil. To znamená, že každá denná hodnota akcie sa vynásobí jej váhou v portfóliu. Tento výpočet je znázornený na obrázku 25.

Obrázok 25: Výpočet historickej hodnoty portfólia

```
if not stock_data_5y.empty:
    if portfolio_history is None:
        portfolio_history = stock_data_5y * allocation
        portfolio_daily_returns = daily_returns * allocation
    else:
        portfolio_history += stock_data_5y * allocation
        portfolio_daily_returns += daily_returns * allocation
```

Zdroj: Vlastné spracovanie

Na základe výpočtu je v ďalšom kroku na obrázku 26 vytvorený interaktívny vizuálny graf.

Obrázok 26: Tvorba grafu vývoja portfólia pomocou knižnice Plotly

```
#  Generovanie grafu pre portfólio (5 rokov)
if portfolio_history is not None:
    fig_portfolio = go.Figure()
    fig_portfolio.add_trace(go.Scatter(x=portfolio_history.index, y=portfolio_history,
                                       mode='lines', name="Portfolio Value"))
    fig_portfolio.update_layout(title="Portfolio Performance (5 years)", xaxis_title="Date", yaxis_title="Value ($)")
    portfolio_graph_html = fig_portfolio.to_html(full_html=False)
```

Zdroj: Vlastné spracovanie

V nasledujúcej časti si uvedieme grafy, ktoré slúžia na vizualizáciu základných štatistík a vývoja portfólia. Tieto grafy sú automaticky generované a vložené do HTML šablóny pomocou premennej *portfolio_graph_html*, takže sa používateľovi zobrazia hneď po načítaní stránky

4.7.1 Vývoj hodnoty portfólia

Zobrazuje historický vývoj kombinovanej hodnoty portfólia, pričom na osi x sú dátumy a na osi y celková hodnota portfólia v $. Ide o hlavnú vizualizáciu celkového vývoja investícií.

Obrázok 27: Graf vývoju hodnoty portfólia za posledných 5 rokov

Zdroj: Vlastné spracovanie

4.7.2 *Vývoj jednotlivých akcií*

V druhom grafe sú zobrazené ceny jednotlivých titulov (akcií a ETF fondov) počas rovnakého obdobia – t.j. 5 rokov. Každý titul je vykreslený vlastnou farebnou čiarou, čo používateľovi umožňuje porovnať vývoj cien medzi jednotlivými titulmi.

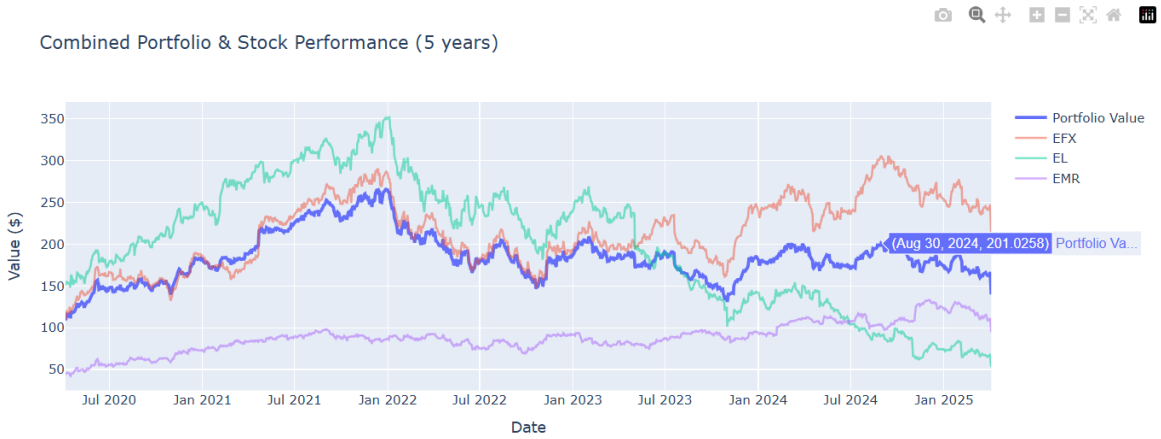
Obrázok 28: Graf vývoju cien jednotlivých akcií za posledných 5 rokov

Zdroj: Vlastné spracovanie

4.7.3 *Kombinovaný graf*

Kombinovaný graf spája predchádzajúce dva grafy – zobrazuje vývoj portfólia spolu s vývojom jednotlivých akcií. Hrubšou čiarou je znázornená hodnota portfólia, pričom jednotlivé akcie sú zobrazené s nižšou priehľadnosťou. Takto vytvorený graf umožňuje jednoducho vyhodnotiť, ktoré akcie najviac prispeli k rastu alebo poklesu portfólia.

Obrázok 29: Graf vývoju cien akcií a portfólia za posledných 5 rokov



Zdroj: Vlastné spracovanie

Grafy sa dynamicky vytvárajú len vtedy, ak sú k dispozícii historické údaje pre všetky vybrané akcie. Ak niektoré chýbajú, používateľ je o tom informovaný a graf sa nevytvorí. Tieto grafy zohrávajú dôležitú úlohu pri spätnej analýze výkonnosti portfólia a pomáhajú identifikovať potenciálne riziká alebo vývojové trendy.

4.8 Uloženie portfólia do systému

Po záverečnom potvrdení portfólia je potrebné uložiť všetky jeho súčasti do databázy. Táto operácia zabezpečuje, že používateľ bude mať prístup k svojmu portfóliu aj po opätovnom prihlásení do systému. Uloženie sa realizuje v rámci route @bp.route("/save_portfolio"), pričom samotná funkcionalita je zabezpečená metódou save_portfolio(). Pri ukladaní sa vykonáva niekoľko krokov:

4.8.1 *Získanie používateľských údajov a dát zo session*

Najprv sa zistí ID aktuálne prihláseného používateľa pomocou current_user.id. Následne sa z pamäte session získa slovník alokácií (allocations)

a vypočítané investované sumy (stock_investments), ktoré boli vytvorené v predchádzajúcich krokoch. Tento proces je znázornený na obrázku 30.

Obrázok 30: Získanie údajov o používateľovi a investíciách zo session

```python
@bp.route("/save_portfolio", methods=["POST"])
@login_required
def save_portfolio():
    user_id = current_user.id
    allocations = session.get("allocations", {})
    stock_investments = session.get("stock_investments", {})
```

Zdroj: Vlastné spracovanie

4.8.2 *Priradenie dátumu a historických cien*

Dôležitou súčasťou pri ukladaní portfólia je aj zachytenie historickej ceny každého titulu k presnému momentu jeho vytvorenia. Použitím knižnice yfinance sa pre každý ticker stiahnu historické ceny za obdobie 5 rokov. Pomocou funkcie *asof()* sa následne nájde najbližší dátum ku dňu vytvorenia portfólia a z neho sa načíta upravená zatváracia cena (tzv. Adjusted Close). Tieto ceny sú následne uložené ako slovník historical_prices.

Obrázok 31: Získavanie historických cien akcií k dátumu vytvorenia portfólia

```python
creation_date = datetime.now(timezone.utc)

historical_prices = {}
for ticker in allocations.keys():
    stock = yf.Ticker(ticker)
    historical_data = stock.history(period="5y")

    if not historical_data.empty and "Adj Close" in historical_data.columns:
        try:
            nearest_date = historical_data.index.asof(creation_date)
            if pd.notna(nearest_date):
                old_price = historical_data.loc[nearest_date]["Adj Close"]
                historical_prices[ticker] = old_price
            else:
                print(f"⚠ {ticker}: No historical data found for {creation_date.strftime('%Y-%m-%d')}")
                historical_prices[ticker] = "N/A"
        except Exception as e:
            print(f"❌ ERROR for {ticker}: {e}")
            historical_prices[ticker] = "N/A"
    else:
        print(f"⚠ {ticker}: No 'Adj Close' data found")
        historical_prices[ticker] = "N/A"
```

Zdroj: Vlastné spracovanie

4.8.3 *Vytvorenie alebo aktualizácia databázového záznamu*

Na obrázku 32 systém najprv overí, či už používateľ má vytvorené portfólio. Ak áno, existujúci záznam sa aktualizuje. Ak portfólio ešte neexistuje, vytvorí sa nový záznam pomocou modelu Portfolio.

Obrázok 32: Uloženie alebo aktualizácia portfólia v databáze

```
portfolio = Portfolio.query.filter_by(user_id=user_id).first()
if portfolio:
    portfolio.allocations = json.dumps(allocations)
    portfolio.stock_investments = json.dumps(stock_investments)
    portfolio.initial_value = 10000.0
    portfolio.current_value = 10000.0
    portfolio.created_at = creation_date
    portfolio.historical_prices = json.dumps(historical_prices)
else:
    portfolio = Portfolio(
        user_id=user_id,
        allocations=json.dumps(allocations),
        stock_investments=json.dumps(stock_investments),
        initial_value=10000.0,
        current_value=10000.0,
        created_at=creation_date,
        historical_prices=json.dumps(historical_prices)
    )
    db.session.add(portfolio)

db.session.commit()

flash("Portfolio successfully saved with historical prices!", "success")
```

Zdroj: Vlastné spracovanie

Model Portfolio obsahuje nasledovné kľúčové informácie, ktoré sa ukladajú pre každého používateľa:

- ID používateľa
- Alokácie a investované sumy (uložené ako JSON)
- Počiatočná a aktuálna hodnota portfólia
- Dátum vytvorenia
- Historické ceny

Tieto údaje sa následne zapíšu do databázy pomocou *db.session.commit()*.

Obrázok 33: Definícia databázového modelu Portfolio v SQLAlchemy

```python
class Portfolio(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    allocations = db.Column(db.Text, nullable=False)
    stock_investments = db.Column(db.Text, nullable=False)
    initial_value = db.Column(db.Float, nullable=False)
    current_value = db.Column(db.Float, nullable=False)
    created_at = db.Column(db.DateTime, default=lambda: datetime.now(timezone.utc))
    historical_prices = db.Column(db.Text, nullable=False)
```

Zdroj: Vlastné spracovanie

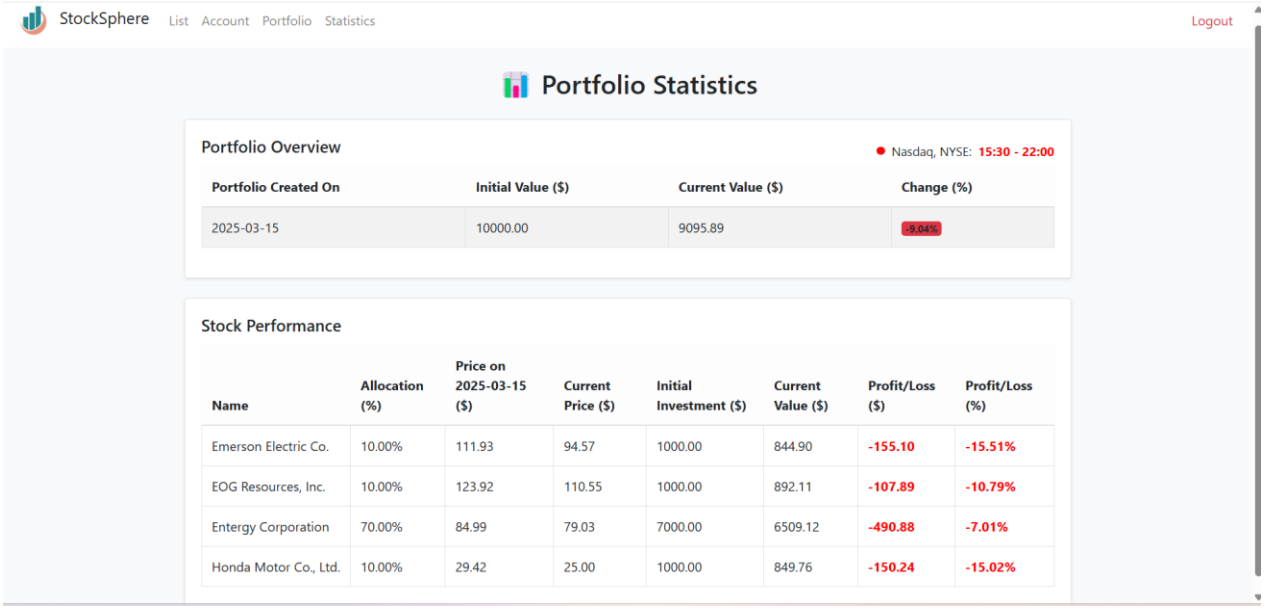
4.8.4 Presmerovanie a spätná väzba

Po úspešnom uložení portfólia sa používateľovi zobrazí informačná hláška (flash message), že portfólio bolo uložené, a je presmerovaný na podstránku s jeho portfóliom pomocou príkazu *redirect(url_for("routes.portfolio")).*

4.9 Zobrazovanie štatistík a analýza dát

Po prihlásení má používateľ prístup do sekcie „Statistics“, ktorá poskytuje detailný prehľad o vývoji jeho investičného portfólia. V tejto podstránke sa zobrazuje porovnanie počiatočnej a aktuálnej hodnoty portfólia, ako aj analýza ziskov a strát jednotlivých akcií.

Obrázok 34: Zobrazenie štatistických údajov portfólia a akcií



Zdroj: Vlastné spracovanie

Na pozadí obrázku 34 funguje route *@bp.route("/stats")*, kde sa z databázy získa uložené portfólio konkrétneho používateľa. V prípade, že portfólio ešte nebolo vytvorené, je používateľ presmerovaný späť na domovskú stránku s varovnou hláškou.

Z databázy sa najprv získa dátum vytvorenia portfólia, počiatočná hodnota a alokácie jednotlivých titulov. Následne sa pre každú zvolenú akciu stiahnu dve kľúčové hodnoty. Historická cena, ktorá zodpovedá dátumu vytvorenia portfólia, a aktuálna cena z trhu. Na základe týchto údajov sa pre každý titul automaticky vypočíta zisk alebo strata v absolútnych hodnotách (Profit/Loss $) a v percentách (Profit/Loss %) ako môžeme vidieť na obrázku 33. Zároveň sa určí aj aktuálna hodnota investície, pričom výpočet prebieha podľa definovaného vzorca:

$$H_{aktualna} = \frac{I}{C_{pociatocna}} \times C_{aktualna} \tag{4.3}$$

Takto vypočítané údaje sú následne uložené do premennej *stock_performance*, ktorá sa prenáša do šablóny na vizuálne zobrazenie. Tento proces je znázornený na obrázku 35.

Obrázok 35: Výpočet aktuálnej hodnoty investície a ziskov/strát na strane servera

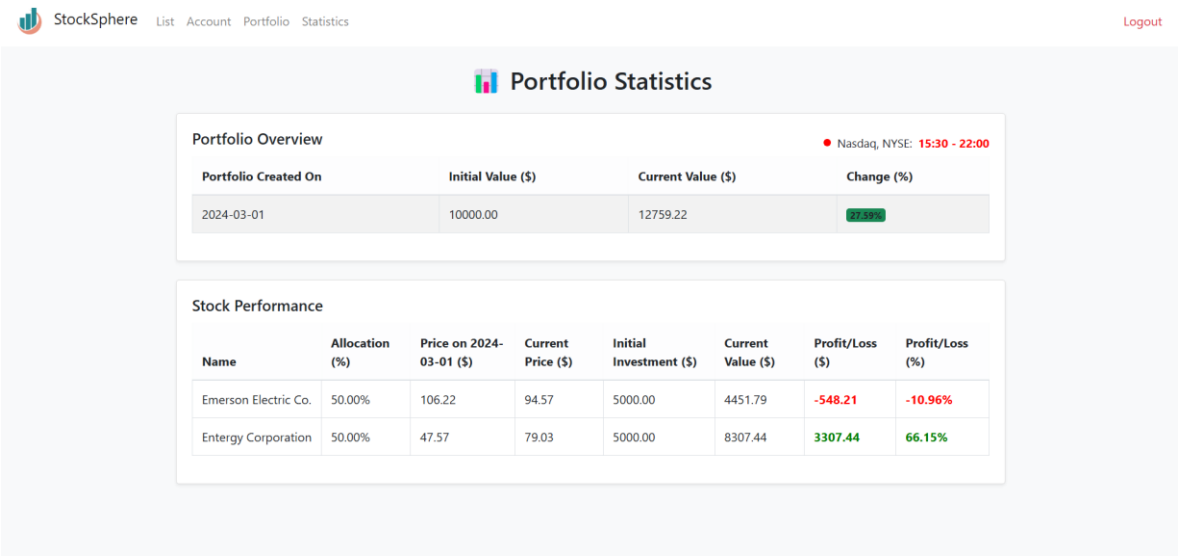
```
# Aktuálna cena
current_data = stock.history(period="1d")
current_price = current_data["Close"].iloc[-1] if not current_data.empty else None

invested_amount = (allocation / 100) * initial_value

if current_price and old_price:
    new_value = (invested_amount / old_price) * current_price
    absolute_profit = new_value - invested_amount
    percent_profit = (absolute_profit / invested_amount) * 100
else:
    new_value, absolute_profit, percent_profit = "N/A", "N/A", "N/A"

stock_performance[ticker] = {
    "name": name,
    "allocation": round(allocation, 2),
    "old_price": round(old_price, 2) if old_price else "N/A",
    "current_price": round(current_price, 2) if current_price else "N/A",
    "old_value": round(invested_amount, 2),
    "new_value": round(new_value, 2) if new_value != "N/A" else "N/A",
    "absolute_profit": round(absolute_profit, 2) if absolute_profit != "N/A" else "N/A",
    "percent_profit": round(percent_profit, 2) if percent_profit != "N/A" else "N/A"
}

if new_value != "N/A":
    current_value += new_value
```

Zdroj: Vlastné spracovanie

4.9.1 *Príklad vývoja portfólia v dlhšom časovom horizonte*

Na obrázku 36 bol pre ilustráciu potenciálneho zhodnotenia portfólia v čase vytvorený modelový príklad, v ktorom bolo portfólio vytvorené pred niekoľkými mesiacmi. Tento prístup simuluje reálny dlhodobejší investičný horizont a umožňuje lepšie vizuálne zachytiť zmenu hodnoty jednotlivých akcií, ako aj celkového portfólia. V tomto prípade je zreteľne viditeľný nárast hodnoty a pozitívne výnosy, ktoré by investor dosiahol v prípade takéhoto vývoja trhu.

Obrázok 36: Zobrazenie dlhodobých štatistických údajov portfólia a akcií



Zdroj: Vlastné spracovanie

5 Diskusia

Výsledkom tejto bakalárskej práce je plne funkčná webová aplikácia s názvom *StockSphere*, ktorej hlavným cieľom je prepojiť teoretické poznatky z oblasti investovania s praktickým modelovaním investičných rozhodnutí. Aplikácia je určená najmä pre začiatočníkov a slúži ako podporný nástroj na pochopenie základných princípov tvorby a správy investičného portfólia.

Používateľ si zostaví vlastné investičné portfólio tak, že si z databázy vyberie akcie a ETF fondy, určí ich váhové zastúpenie a následne sleduje, ako sa portfólio správa v čase na základe reálnych trhových údajov. V rámci aplikácie je implementovaných niekoľko základných funkcií, ktoré používateľovi umožňujú aktívne pracovať s investičným portfóliom. Používateľ má možnosť vybrať si investičné nástroje – konkrétne akcie a ETF fondy – z databázy, ktorá obsahuje aj doplňujúce informácie o sektore, krajine pôvodu a názve spoločnosti. Ceny týchto titulov sú dynamicky získavané z verejne dostupného zdroja Yahoo Finance prostredníctvom knižnice yfinance.

Na základe výberu a zadaného investičného rozpočtu sa vypočítavajú alokácie a simulované investované sumy. Aplikácia zároveň ukazuje, ako sa portfóliu darí, a to pomocou výpočtov ako ročný výnos (YTD), či kolísavosť cien jednotlivých akcií aj celého portfólia. Tieto údaje pomáhajú používateľovi získať lepší obraz o rizikovosti investícií. Na podporu prehľadnosti sú zobrazené aj grafy, ktoré znázorňujú, ako je portfólio rozdelené podľa sektorov a krajín. Vďaka tomu si môže používateľ jednoducho všimnúť, do ktorých oblastí investuje viac a kde má prípadné medzery v diverzifikácii. Každé portfólio sa po vytvorení uloží do databázy, aby si ho mohol používateľ neskôr zobraziť alebo upraviť. O ukladanie sa stará nástroj SQLAlchemy, ktorý zjednodušuje prácu s databázou. V porovnaní s dostupnými investičnými nástrojmi, ktoré sú často zamerané na reálne obchodovanie alebo pokročilé technické analýzy, ponúka aplikácia *StockSphere* zjednodušené a prehľadné rozhranie s dôrazom na edukačný rozmer. Používateľ nepotrebuje zadávať vlastné API kľúče ani sa registrovať do burzových platforiem. Všetky potrebné údaje sú získavané automatizovane a vizualizované priamo v prehliadači.

V rámci diskusie o dosiahnutých výsledkoch možno poukázať na niekoľko aspektov. Aplikácia sa ukázala ako spoľahlivá pri načítavaní trhových údajov. Vo väčšine prípadov fungovalo získavanie cien bez problémov. V niektorých prípadoch, najmä pri menej obchodovaných tituloch, sa však vyskytli výpadky alebo nedostupnosť údajov. V takýchto prípadoch systém reaguje upozornením a namiesto chýbajúcich údajov zobrazí informatívnu hodnotu, napríklad „N/A“, aby používateľ vedel, že dané dáta momentálne nie sú dostupné. Presnosť výpočtov výkonnostných metrík, ako sú ročné výnosy (YTD), volatilita či celkový zisk alebo strata, bola testovaná na viacerých vzorových portfóliách. Výsledky sa ukázali byť konzistentné a zhodné s údajmi, ktoré poskytujú bežne používané finančné nástroje. Z hľadiska používateľskej skúsenosti aplikácia pôsobí jednoducho a prehľadne. Vďaka využitiu Bootstrap dizajnu je orientácia v rozhraní jednoduchá, pričom stránkovanie, validácia formulárov a spätná väzba pri chybách boli implementované tak, aby používateľ vždy vedel, čo sa v aplikácii deje a ako má ďalej postupovať.

Medzi aktuálne obmedzenia aplikácie patrí napríklad absencia dividend a ich reinvestície. Používateľ má momentálne možnosť spravovať len jedno portfólio a aplikácia zatiaľ neumožňuje sledovanie vývoja v rôznych menách. Tieto oblasti však predstavujú potenciál pre ďalší vývoj. V porovnaní s podobnými riešeniami (napr. aplikácie ako *PortfolioVisualizer* alebo *Yahoo Finance Portfolio*) je výhodou tejto aplikácie jej open-source charakter, jednoduchosť a sústredenie na konkrétny trh používateľa. Navyše, jej rozšíriteľnosť umožňuje budúce vylepšenia ako napr. pridanie grafov pre jednotlivé sektory v čase, notifikácie o výkyvoch trhu, alebo export portfólia do PDF.

Z pohľadu ďalšieho rozvoja aplikácie by bolo vhodné zvážiť rozšírenie o započítanie transakčných nákladov spojených s nákupom a predajom cenných papierov. Tieto náklady môžu byť najmä pri menších objemoch investícií výrazné a majú priamy vplyv na reálnu výkonnosť portfólia. Výška transakčných poplatkov sa navyše líši v závislosti od použitej platformy napríklad maklérske spoločnosti ako Lynx alebo Interactive Brokers účtujú približne 0,12 % z objemu obchodu, zatiaľ čo platformy ako eToro si účtujú fixný poplatok vo výške 1 USD a zároveň pracujú s vyšším spreadom.

Z pohľadu autora bola najväčšou výzvou práve synchronizácia front-endovej a back-endovej logiky - teda zabezpečiť, aby všetky zmeny používateľa v rozhraní (napr. výber

akcií, zadanie alokácií) boli správne použité vo výpočtoch a následne uložené do databázy. Tento problém sa podarilo vyriešiť vďaka prehľadnej štruktúre session a oddeleným Flask routom.

Výsledná webová aplikácia ukazuje, že aj s využitím jednoduchších technológií sa dá vytvoriť nástroj, ktorý môže slúžiť ako pomôcka pri osvojovaní si základných investičných princípov. Cieľom bolo priblížiť investovanie zrozumiteľne a prakticky, bez potreby reálnych peňazí. V praxi môže slúžiť ako prostriedok pri výučbe finančnej gramotnosti, investičných základov alebo tvorby webových riešení. Navyše je pripravená na ďalšie rozšírenia.

Záver

Cieľom tejto bakalárskej práce bolo navrhnúť a implementovať webovú aplikáciu slúžiacu ako edukačný nástroj pre jednotlivcov so záujmom o investovanie do akcií. Hlavným zámerom bolo vytvoriť používateľsky prístupné prostredie, ktoré umožní praktické zostavovanie investičného portfólia, jeho sledovanie a analýzu rizikových a výnosových charakteristík jednotlivých titulov.

Aplikácia bola zostrojená s dôrazom na zrozumiteľnosť, a to aj pre úplných začiatočníkov bez predchádzajúcich skúseností s investovaním. Tento aspekt možno považovať za jej najväčší prínos. V porovnaní s existujúcimi nástrojmi kladie táto aplikácia dôraz predovšetkým na vzdelávací rozmer a praktické využitie.

Realizácia práce predstavovala cennú príležitosť na rozvoj technických zručností, konkrétne v oblasti integrácie technológií ako Flask, SQLAlchemy, yFinance a Plotly. Zároveň umožnila hlbšie pochopenie celého vývojového procesu webovej aplikácie od počiatočného návrhu až po finálnu implementáciu.

Do budúcna existuje viacero smerov, ktorými by sa aplikácia mohla ďalej rozvíjať. Medzi najbližšie možnosti patrí pridanie funkcií, ako je sledovanie dividend, optimalizácia portfólia či automatické aktualizovanie údajov. Rovnako by sa dali rozšíriť nástroje na analýzu výkonnosti investícií, čím by aplikácia poskytovala ešte komplexnejší pohľad na spravované portfólio. Z praktického hľadiska môže mať využitie aj v oblasti vzdelávania napríklad ako pomôcka pri výučbe finančnej gramotnosti alebo základov investovania. Možnosťou je aj vytvorenie otvorenej online verzie, ktorá by bola dostupná širokej verejnosti. Na záver možno povedať, že aplikácia vytvorená v rámci tejto práce ponúka solídny základ pre ďalší rozvoj. Aplikácia má potenciál byť praktickým pomocníkom pre každého, kto chce s začať s investovaním.

Zoznam použitej literatúry

Bootstrap. (2024). *Bootstrap 5 Documentation* [online]. Dostupné na: https://getbootstrap.com

CANVAS BUSINESS MODEL. (2025). TradingView SWOT Analysis [online]. Dostupné na: https://canvasbusinessmodel.com/products/tradingview-swot-analysis

FLASK. (2024). *Flask Documentation* [online]. Dostupné na: https://flask.palletsprojects.com

GRANT, Mitchell. (2024). *When Do Stock Market Exchanges Close?* [online]. Investopedia. Dostupné na: https://www.investopedia.com/ask/answers/040115/when-do-stock-market-exchanges-close.asp

HAYES, Adam. (2024). *Portfolio Variance: Definition, Formula, Calculation, and Example* [online]. Investopedia. Dostupné na: https://www.investopedia.com/terms/p/portfolio-variance.asp

HAYES, Adam. (2024). *Volatility: Meaning in Finance and How It Works With Stocks* [online]. Investopedia. Dostupné na: https://www.investopedia.com/terms/v/volatility.asp

LEE, Jinkyoo, KIM, Rein H., YI, Sung-Whan a KANG, Jaewoo. (2020). *MAPS: Multi-agent Reinforcement Learning-based Portfolio Management System*. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20), 4574–4580. [online]. Dostupné na: https://www.ijcai.org/proceedings/2020/0623.pdf

LIBERTO, Daniel. (2025). *Year to Date (YTD): What It Means and How to Use It* [online]. Investopedia. Dostupné na: https://www.investopedia.com/terms/y/ytd.asp

McKINNEY, Wes. (2022). *Python for Data Analysis*. 3. vyd. Sebastopol: O'Reilly Media. ISBN 978-1491957660

Plotly Technologies Inc. (2024). *Plotly Python Open Source Graphing Library* [online]. Dostupné na: https://plotly.com/python

Python Software Foundation. (2025). *Python 3 Documentation* [online]. Dostupné na: https://docs.python.org/3

RAN, Aroussi. (2025). *yFinance Documentation – Yahoo Finance Python Library* [online]. Dostupné na: https://pypi.org/project/yfinance

RED HAT. (2020). *What is a REST API?* [online]. Dostupné na: https://www.redhat.com/en/topics/api/what-is-a-rest-api

REILLY, Frank. K., BROWN, Keith. C. (2012). *Investment Analysis and Portfolio Management*. 10. vyd. Mason, OH: South-Western Cengage Learning. ISBN 978-0-538-48238-7

SMITH, Lisa. (2025). Stock Simulator: A Useful Tool for Investors [online]. Investopedia. Dostupné na: https://www.investopedia.com/articles/stocks/08/stock-simulator.asp

SQLAlchemy. (2025). *SQLAlchemy ORM Documentation – ORM Quick Start* [online]. Dostupné na: https://docs.sqlalchemy.org/en/20/orm/quickstart.html

TARDI, Carla. (2024). *Portfolio*. Investopedia [online]. Dostupné na: https://www.investopedia.com/terms/p/portfolio.asp

Visual Studio Code. (2024). *Visual Studio Code Documentation* [online]. Dostupné na: https://code.visualstudio.com

W3Schools. (2024). *W3Schools HTML Tutorial* [online]. Dostupné na: https://www.w3schools.com/html

W3Schools. (2024). *W3Schools JavaScript Tutorial* [online]. Dostupné na: https://www.w3schools.com/js

W3Schools. (2024). *W3Schools Python Tutorial* [online]. Dostupné na: https://www.w3schools.com/python

Yahoo Finance. (2025). *Yahoo Finance Homepage* [online]. Dostupné na: https://finance.yahoo.com