|  |  |  |
| --- | --- | --- |
|  | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Finanční webová aplikace** | | |
| František Švec | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2016/2017 | |

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité   
informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové   
a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2020

*podpis autora práce*

**ANOTACE**

Cílem projektu bylo vytvořit funkční webovou aplikaci, která má shromažďovat data týkající se akcií, různých komodit a měn, se kterými se běžně obchoduje na finančním trhu, tyto informace má následovně upravovat a vytvářet s nich porovnání, grafy a další věci. Tyto data následně vyobrazuje na webové stránce většinou do přehledných tabulek, ve kterých se snadno dá orientovat.

Tato webová aplikace je vytvořena ve frameworku Django, získávání dat je řešeno pomocí externích knihoven jazyka python, pro vyobrazení dat např. javascriptová knihovna Chart.js a pro ukládaní dat je využívána RESTapi, a databáze Sqlite3.

OBSAH

[Úvod 4](#_Toc60962402)

[1 Teoretická a metodická východiska 5](#_Toc60962403)

[1.1 Online obchodování a podobné weby 5](#_Toc60962404)

[1.2 Problematika získávání dat 5](#_Toc60962405)

[2 Využité technologie 6](#_Toc60962406)

[2.1 Framework Django 6](#_Toc60962407)

[2.2 Django REST framework 6](#_Toc60962408)

[2.3 Chart.js 6](#_Toc60962409)

[2.4 Pandas dataframe 6](#_Toc60962410)

[2.5 Pandas datareader 7](#_Toc60962411)

[2.6 Beautiful Soup 7](#_Toc60962412)

[2.7 Alpha-vantage 7](#_Toc60962413)

[3 Způsoby řešení a použité postupy 8](#_Toc60962414)

[3.1 Získávaní dat 8](#_Toc60962415)

[3.1.1 Webscraping 8](#_Toc60962416)

[3.1.2 Získávaní historických dat z API 9](#_Toc60962417)

[3.1.3 Získávaní dat v reálném čase z API 10](#_Toc60962418)

[3.2 Tvoření vlastního API 11](#_Toc60962419)

[3.3 Přistupování k datům z Rest API v JS souboru 12](#_Toc60962420)

[3.4 Zobrazovaní dat 12](#_Toc60962421)

[3.4.1 Zobrazování dat do grafů 12](#_Toc60962422)

[3.5 Přihlašování 13](#_Toc60962423)

[4 VÝSTUPY PRÁCE 14](#_Toc60962424)

[4.1 API 14](#_Toc60962425)

[4.2 Tabulky 14](#_Toc60962426)

[4.3 Grafy 15](#_Toc60962427)

[Závěr 16](#_Toc60962428)

Úvod

K tvorbě informační webové aplikace o finančním trhu, a přehledů jak o různých akciích, tak zahraničními měnami, nebo komoditami se kterými se obchoduje, mě vedlo spoustu věcí, asi ta nejzásadnější, byla absence podobných projektů nebo webových aplikací, a přišlo mi zajímavé se touto problematikou zabývat. Další motivací pro vytvoření tohoto projektu, byla příležitost se naučit nové dovednosti v oboru, získávaní a zpracování dat. Webová aplikace by měla sloužit pro sloučení dat ohledně obchodování s akciemi, a dalšími věcmi podobného rázu, pro rychlí přehled o situaci na trhu sloučenou do přehledně upravené webové stránky, a to každý den s novými daty.

Hlavní cíl projektu bylo tedy vytvořit funkční webovou stránku, ta je vytvořena ve frameworku Django. Dalším cílem bylo zajistit co největší objem dat, aby tato stránka byla co nejpoužitelnější, data jsou hlavním stavebním kamenem této webové aplikace, a byl to obzvlášť nelehký úkol zajistit jejich dostatek. Tyto data jsou posílány na API, ze kterého jsou poté využívány na webových stránkách. Tuto metodu jsem zvolil, také proto že v budoucnu by tohle API, mohlo být přístupné pro ostatní uživatele, kteří by chtěli využívat tyto data, samozřejmě například historická data se zároveň ukládají do databáze, protože u nich není předpoklad žádné změny. Další cíl bylo zajistit nějaké obohacení těchto stránek, to jsem zvolil ve formě útržků zpráv ze světa finančnictví.

V této dokumentaci podrobně popisuji úskalí a problémy získávání dat, a jak dále s těmito daty můžete nakládat.

# Teoretická a metodická východiska

## Online obchodování a podobné weby

V dnešní době začíná být stále více populární obchodovat s různými akciemi(cennými papíry), komoditami, nebo zahraničními měnami či kryptoměnami. Tento trend dává za vznik spousty e brokerů(poskytovatel služeb v podobě prodeje a nákupu výše zmíněných věcí, prostřednictvím digitální platformy), což dělá obchodování lehčí než kdy dřív, zmíněné e-broker stránky vám sice poskytnou tyto informace, ale zároveň jsou to oni od kterých nakupujete a prodáváte, a je dobré mít pro srovnání nějaký nezávislí zdroj informací a nejlépe stránku , která by tyto informace získávala z více zdrojů, pro co největší. Na českém internetu taková stránka existuje, jsou to kurzy.cz , ačkoli má tato stránka taky pár svých problémů, tak i přesto mi byla relativně velkou inspirací, ostatně z ní čerpám některé data. Ze zahraničních stránek je to Yahoofinance, je to jedno z největších finančních medií na světě.

## Problematika získávání dat

Jak je již výše zmíněno pro co největší objektivitu těchto stránek bylo potřeba získat co největší objem dat, což se ukázalo jako relativně velký problém. Pro prozkoumáni různých možností, jak získávat tyto data, jsem přišel na to že neexistuje zas tak velký počet otevřených API , po ještě bližším přezkoumaní jsem přišel na to že většina lepších API mají omezený počet requestů, a pro zvýšení počtu requestů musíte zaplatit, a ty zbylé API které mají velký počet requestů a nemusíte za ně plati, nabízí ve směs ty samé data. Proto jsem se rozhodl pro získáni některých dat vyžití metody webscraping(vysvětleno níže), tato metoda v podstatě vyřešila všechny problémy se získáváním dat. Jediný problém této metody je že vždycky nemusí být v souladu s podmínkami stránkami ze které data získáváte.

# Využité technologie

## Framework Django

Pro vytvoření webové aplikace, jsem se rozhodl použít framework Django. Je napsaný v Pythonu a podporuje ty nejmodernější technologie, obsahuje již spoustu zabudovaných technologií, a má velice rozsáhlé možnosti. Staví na principech maximální produktivity a znovu použitelnosti. Neustále se vyvíjí a začíná se stávat populárním i mezi českými webovými vývojáři. Rozhodl jsem se také použít kvůli přítomnosti jazyku Python, který je velice příhodný pro tvorbu věcí spojených s  vývojem této aplikace.

## Django REST framework

Django REST framework je silný a flexibilní nástroj pro vytváření webových API. Do tohoto API ukládám většinu získaných dat. A pak k nim přistupuju pomocí metody jQuery AJAX. Django REST framework, také nabízí serializaci dat která podporuje jak ORM tak non-ORM datové zdroje.

## Chart.js

Pro zobrazení grafů jsem si vybral javascriptovou knihovnu Chart.js je to velmi populární knihovna. Tato knihovna má nespočet možností jak můžou grafy vypadat od klasických čárových grafů s jedním atributem, přes koláčové grafy s více atributama až po komínové grafy. Jediný problém je že se grafy vykreslují do prvku <canvas>, takže s responzivitou to je horší. Šířkou a výškou se ale grafy přizpůsobovat umí.

## Pandas dataframe

Pandas dataframe je jedna z nejoptimálnějších a nejpoužívanějších datových struktur pro ukládaní dat o akciích. Základním stavebním kamenem knihovny Pandas je přitom typ Series(datová řada), který zapouzdřuje jednodimenzionální pole z knihovny Numpy. Datová řada představuje uspořádaný sloupec údajů, které mají shodný typ(například int64 nebo float64 atd.), přičemž každému prvku je přiřazen index.

## Pandas datareader

Je podknihovna Pandas, tato knihovna umožnuje vytvořit dataframe z různých internetových zdrojů, já tuto knihovnu konkrétně využívám k získávání historických dat ze stránky Yahoo Finance. Tato metoda je o dost pohodlnější než získávat data z API, pomocí reqeustů.

## Beautiful Soup

Knihovna Beautiful Soup, je nástroj pro parsování a získávání dat z HTML nebo XML k**ó**dů. Tato metoda umožnuje získávat obsah z webových stránek bez HTML značek, jako například čistý text. Jelikož tento způsob získávání dat využívám pro svou aplikaci velice často, protože pro určité data které jsem pro svou aplikaci potřeboval získat jsem nenašel žádné API. Tak bylo na místě dobře zvážit kterou knihovnu pro tento způsob získávání dat použít, protože existuje ještě jedna python knihovna, stejně hojně používána jako Beatiful Soup jménem Selenium , která slouží pro získávání dat tohoto typu, ale ta je o dost komplexnější, a proto jsem zvolil knihovnu Beautiful Soup, která je o dost jednoduší, ale při případném dalším rozvoji tohoto projektu , bych určitě zvážil přechod na knihovnu Selenium.

## Alpha-vantage

Alpha-vantage je knihovna, která umožnuje přístup k datům o akciích v realném čase. Má ale jeden zásadní problém a to maximální počet requestů 5 za minutu, což je velice limitující, jinak tato knihovna disponuje spoustou různých užitečných nástrojů.

## 

# Způsoby řešení a použité postupy

## Získávaní dat

Tohle byl největší problém k řešení na tomto projektu jak je již výše zmíněno.

### Webscraping

Většina dat kterou používám pro svoji aplikaci je získávána metodou web scraping, nebo web harvesting je způsob získávání dat přímo z webových stránek.

**Přistupování ke stránkám**

Ke stránkám ze kterých čerpám data přistupuji pomocí knihovny requests, při přístupu ke stránce přidávám do requestu User agent, tento string posílám společně z requestem na danou stránku. Přítomnost User agenta je velice důležitá, protože jeho absence by mohla způsobit, že stránka budu registrovat moji přítomnost jako bota, a mohli by mi zablokovat přístup k stránkám.

USER\_AGENT = "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/44.0.2403.157 Safari/537.36"

LANGUAGE = "en-US,en;q=0.5"

session = requests.Session()

session.headers['User-Agent'] = USER\_AGENT

session.headers['Accept-Language'] = LANGUAGE

session.headers['Content-Language'] = LANGUAGE

source = session.get('https://www.reuters.com/news/archive/businessNews').text

soup = BeautifulSoup(source, 'lxml')

**Parsování a vyhledávaní**

Vyhledávaní provádím pomocí metody find\_all() nebo find() z knihovny Beautiful Soup.

Tyto obě metody potřebují argument ve podobě HTML tagu, dá se taky použít zároveň s argumentem class\_ , který vyhledá určitý HTML tag s touto classou, tento způsob používám nejčastěji. Metodou find\_all() lze najít všechny tagy div následně projít cyklem for a dostat z nich metodou .text jen holý text. Níže můžete vidět příklad jednoduchého k**ó**du na získání úryvků zpráv. V cyklu for se nachází try a except funkce, protože metoda find\_all a find občas vrací datový typ Nonetype, když najde například prázdný řádek, a potom dochází k erroru, protože metoda text, koliduje s datovým typem Nonetype, a funkce try a except dokáže tento error snadno odstranit.

list = []

**for** article **in** soup.find\_all('article',class\_='story'):

    zaklad = article.find('div',class\_= 'story-content')

**try**:

        data ={

        'nadpis':zaklad.h3.text,

        'texty':zaklad.p.text,

        }

        list.append(data)

**except** Exception as e:

**pass**

**Získávání dat z tabulek**

Funguje na úplně stejném principu jako získávání úryvků, ukládaní dat je řešeno v cyklu for, ve kterém se vytváří slovník, který se po každém projetí řádku přidáva do listu, vytvoří se tak list slovníků, ke kterému se dá přistupovat pomocí indexů.

**for** tabulka **in**  soup.find\_all('table',class\_='pd pdw'):

    rows = tabulka.find\_all('tr',class\_='ps')

**for** row **in** rows:

        data = {

        'nazev\_meny':row.find\_all('td')[0].text,

        'zkratka':row.find\_all('td')[2].text,

        'pocet':row.find\_all('td')[3].text,

        'datum1':row.find\_all('td')[4].text,

        'kurz':row.find\_all('td')[5].text,

        'datum2':row.find\_all('td')[6].text,

        }

        list.append(data)

Podobně jako tahle funkce, vypadají i ostatní funkce pro získávání dat, jelikož většina informací o akciích se nachází v tabulkách.

Přístup k těmto datům vypadá takto:

data = list[0]["nazev"]

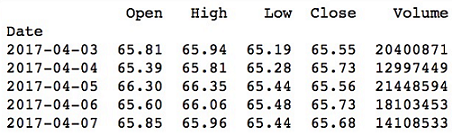
### Získávaní historických dat z API

Přístup k těmto datum zajišťuji pomocí podknihovny pandas a to pandas-datareader, funguje to tak že do metody DataReader dosadím 4 atributy, první je tzv. symbol ackie, druhý je web z API, protože pandas-datareader jich podporuje víc, v mém případě je to Yahoo, další je datum od kterého data chci ,a druhé datum do kterého data chci. Vypadá to takhle:

web = pandas\_datareader

data = web.DataReader('MSFT', 'yahoo', '2021-01-05', '2021-01-0')

Tato proměnná potom vrací pandas dataframe, který vypadá takto:



Já používám pro přístup k těmto datům používám cyklus for, kde jako iterátor dosadím list Symbolů, na který použiju metodu enumarate(), abych jím mohl iterovat. Poté dozasuje místo symbolu ackie i, a k proměnné z indexem i přiřazuji proměnnou Datareader() s indexem název indexu např. ['Close'].

symboly = ['SPY', 'AAPL', 'MSFT', 'HOG', 'INTC', 'T', 'WMT]

**def** data(strt, end, symboly\_list, this\_price):

    data\_ackie = pd.DataFrame([])

**for** idx, i **in** enumerate(symboly\_list):

**try**:

            total = web.DataReader(i, 'yahoo', strt, end)

            data\_ackie[i] = total[this\_price]

**except** Exception as e:

**pass**

**return** data\_ackie

Přistupovaní k těmto datům vypadá takto:

dfT = data(tyden, cas\_dnes, symboly, 'Close')

data = yahoo\_finance.dfR['AAPL']

### Získávaní dat v reálném čase z API

Pro tento typ dat používám instalovatelnou knihovnu Alpha-vantage, data získávam tak, že vytvořím proměnnou ve které je metoda Timeseries, která musí obsahovat 2 atributy, a to api\_key a výstupní formát. Poté s touto vytvořenou proměnou použiji metodu get\_interday, která potřebuje atribut symbol akcie, interval a outpusize.

**def** real(stock):

api\_key = 'GQ43JWXBO74Y0Z9N'

ts = TimeSeries(key=api\_key, output\_format='pandas')

df = ts.get\_intraday(stock.upper(), interval='1min', outputsize='compact')

**return** df

## Tvoření vlastního API

Vytvoření zajištují pomocí django rest framework. Api vytvářím tak že vytvořím class APIview, který se o všechno postará, jak o povolení přístupu uživatelům k tomuto API, tak o autentifikaci těchto přistupujících uživatelů. Data vkládám do API v podobě python slovníků.

**class** KurzyData(APIView):

**def** get(self, request, format=None):

        aud = {

            "nazev": kurzy.list[0]["nazev\_meny"],

            "zkratka": kurzy.list[0]["zkratka"],

            "pocet": kurzy.list[0]["pocet"],

            "datum1": kurzy.list[0]["datum1"],

            "kurz": kurzy.list[0]["kurz"],

            "datum2": kurzy.list[0]["datum2"],

        }

## Přistupování k datům z Rest API v JS souboru

Pro přístup k datům ze svojí vlastní API používám Ajax, jako přístupovou metodu.

**var** endpoint = '/api/komodity-data'

$.ajax({

    method: "GET",

    url: endpoint,

    success: **function**(data){

       komodita = data

    },

    error: **function**(error\_data){

        console.log("error")

        console.log(error\_data)

    }

})

Přístup k těmto datům je potom velice jednoduchý, v praxi to vypadá takhle:

data = komodita.energie.topnyolej

## Zobrazovaní dat

### Zobrazování dat do grafů

Pro zobrazovaní dat do grafů využívám knihovnu Chart.js, pro vykreslení grafu je potřeba vytvořit canvas v HTML souboru, a v JS souboru potom vytvořit strukturu grafu, a vložit do něj data.

**<canvas** id="canvas-okno1" width="800" height="300"**></canvas>**

**var** a = document.getElementById('canvas-okno1');

**var** myChartR = **new** Chart(a, {

          type: 'line',

          data: {

      labels: neco.Tabulka.microsoft.datum,

  datasets: [{

      data:neco.Tabulka.microsoft.data,

      label: neco.Tabulka.microsoft.label,

      borderColor: "#8e5ea2",

      fill: **false**

    }]

  }});

## Přihlašování

Pro možnost přihlášení využívám již zabudované funkce, a předvytvořené věci pro vytvoření uživatele a přihlášení uživatele.

**def** register(request):

**if** request.method =='POST':

        form = CreateUserForm(request.POST)

**if** form.is\_valid():

            form.save()

        user = form.cleaned\_data.get('username')

        messages.success(request, 'Účet byl úspěšně vytvořen pro uživatele: ' + user)

**return** redirect('login')

**else**:

        form = CreateUserForm()

        args = {'form': form}

**return** render(request, 'register.html', args)

**def** login\_page(reqeust):

**if** reqeust.method == 'POST':

        username = reqeust.POST.get('username')

        password = reqeust.POST.get('password')

        user = authenticate(reqeust, username=username, password=password)

**if** user **is** **not** None:

            login(reqeust, user)

**return** redirect('chart')

**else**:

            messages.info(reqeust, 'Špatné jméno nebo heslo!!')

    context = {}

**return** render(reqeust, 'login.html', context)

# VÝSTUPY PRÁCE

Výstupem práce je funkční webová stránka.

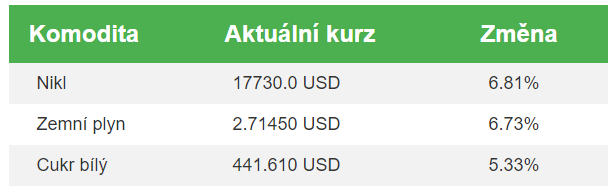
## API

Na snímku níže můžete vidět ukázku z vytvořeného API, ve kterém se nachází kurzy měn, v budoucnu by tohle API, mohli využívat i jiní uživatelé pro svoji potřebu.



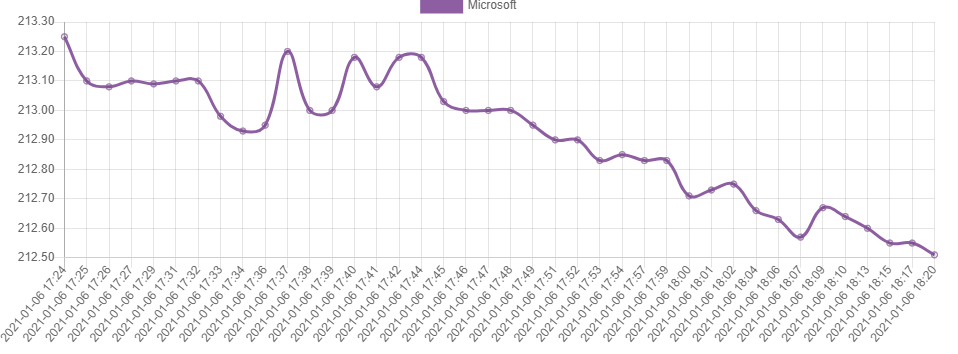
## Tabulky

Většina dat na mé webové aplikaci, je v tabulkách. Tabulky vypadají takto:



## Grafy

Když jsou k určité komoditě nebo akcii přístupné historické data, snažím se využít grafické zobrazení pro lepší představení o vývoji trhu, avšak ne u každé akcie to není možné.



# Závěr

Cílem projektu , bylo vytvořit webovou aplikaci, což si myslím že se relativně povedlo, určitě si myslím že práce na tomto projektu mě hodně naučila a posunula dopředu. Cíle nebyli úplně splněné na 100%, do budoucna je zde spoustu možných vylepšení, jako například vytvoření vlastního watchlistu, protože přihlášení absolutně ztrácí význam.