Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Bakalářská práce

Tvorba webové aplikace pro získávání dat použitelných při výuce zpracovávání dat

Creation of a web application for gaining data that can be used in the lessons of data processing

Vypracoval: Svoboda Martin

Vedoucí práce: Mgr. Václav Šimandl, Ph.D.

České Budějovice 2024

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Pedagogická fakulta

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin SVOBODA**
Osobní číslo:	**P21388**
Studijní program:	**B0114A300110 Oborové studium se zaměřením na vzdělávání na 2. stupni základní školy**
Specializace:	**Anglický jazyk se zaměřením na vzdělávání na 2. stupni ZŠ**
	Informační technologie se zaměřením na vzdělávání na 2. stupni ZŠ
Téma práce:	**Tvorba webové aplikace pro získávání dat použitelných při výuce zpracovávání dat**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování

Cílem práce je vytvořit webovou aplikaci, která bude nabízet aktuální data vhodná pro výuku hromadného zpracování dat. Data budou získávána z vybraných serverů se sportovní tématikou, obsahující vývoj cen komodit na světových trzích a podobně. Uživatel si nejprve vybere požadovaný typ dat (případně jejich konkrétní vzorek) a následně mu bude zobrazen náhled těchto dat. Aplikace bude umožňovat zvolená data stáhnout ve formátu XLSX nebo CSV. Výkonná část aplikace bude vytvořena pomocí jazyka Python; její spolupráce s frontendem bude zajištěna pomocí vhodného frameworku. Přínos práce bude spočívat nejen ve vytvoření pomůcky použitelné při výuce hromadného zpracování dat, ale také v nalezení možností a limitů tvorby webových aplikací za použití jazyka Python.

Rozsah pracovní zprávy:	**40**
Rozsah grafických prací:	**–**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. BIN UZAYR, S. (ed.) Mastering Python for web: a beginner's guide. Boca Raton: CRC Press, 2022. ISBN 978-1-032-13565-6.
2. HEROUT, P. Testování pro programátory. České Budějovice: Kopp, 2016. ISBN 978-80-7232-481-1.
3. PECINOVSKÝ, R. Začínáme programovat v jazyku Python. 2. přepracované a rozšířené vydání. Praha: Grada Publishing, 2022. ISBN 978-80-271--3609-4.
4. PECINOVSKÝ, R. Python: knihovny pro práci s daty pro verzi 3.11. Praha: Grada Publishing, 2023. ISBN 978-80-271-0659-2.
5. SULCAS, A. Python Web Scraping Tutorial: Step-By-Step. Oxylabs [online]. Oxylabs, 2022. Dostupné z: https://oxylabs.io/blog/python-web-scraping
6. WIEGERS, K. E. Požadavky na software: Od zadání k architektuře aplikace. Brno: Computer Press, 2008. ISBN 978-80-251-1877-1.

Vedoucí bakalářské práce: **Mgr. Václav Šimandl, Ph.D.**
Katedra informatiky

Datum zadání bakalářské práce: **11. dubna 2023**
Termín odevzdání bakalářské práce: **30. dubna 2024**

L.S.

doc. RNDr. Helena Koldová, Ph.D.
děkanka

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 11. dubna 2023

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se za chováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 18. dubna 2024

Martin Svoboda

Abstrakt

Cílem práce je vytvoření aplikace, která bude sloužit jako nástroj při výuce hromadného zpracování dat. Aplikace přebírá data z jiných serverů, zpracovává data do čitelné podoby a předkládá data uživateli ke stažení. Dále aplikace zpracovává data do souborů přímo připravených k použití ve výuce. V aplikaci se nachází aktuální data počasí pro Českou republiku, rozdělené do krajů. Sportovní tabulky nejvyšších soutěží v basketbale, fotbale a hokeji. Basketbalová tabulka nabízí i historickou tabulku všech týmů, které se soutěže účastnily. Fotbalová a hokejová tabulka lze rozdělit na domácí a venkovní utkání. Také se zde nachází historie vývoje cen zlata, stříbra, platiny, dvou typů ropy, elektřiny a zemního plynu. Tato data pak budou zobrazena uživatelům a předložena ke stažení ve dvou formátech, CSV a XLSX. Backend aplikace je napsaný v programovacím jazyce Python, za použití frameworku Django. Frontend aplikace je pak napsaný v HTML, CSS a Javascriptu. Aplikace byla otestována na základní škole. Na základě testování jsem opravil chyby pramenící z přístupů více uživatelů současně. Dále se změnil přístup předkládání souborů, nyní se nemusí stahovat pokaždé všechna data, ale pokud byla stažena nedávno, tak se předloží již stažená data. Přínos mé práce spočívá ve zjednodušení přístupu k datům a zrychlení přípravy na výuku. Během své práce jsem nezaznamenal žádné konkrétní limity Pythonu, které by mi znemožnily mojí práci dokončit.

Klíčová slova

Aplikace, Python, Django, data, hromadné zpracování dat, pomůcka pro výuku

Abstract

The goal of this thesis is to create an application, which will be used as a tool in lessons of data processing. The application takes data from other sites, processes them into more suitable format and presents them for the user to download. In the application, there will be up to date data of weather, divided into regions, sports tables of the highest sport leagues in the Czech Republic. Basketball offers a table of all historic teams that participated in the league. Football and hockey tables can be divided into home and away matches. There is also a history of prices of gold, silver, platinum, two types of crude oil, electricity, and natural gas. These data will then be shown to the user and presented for download in two types of files, CSV and XLSX. Backend of the application is written in programming language Python with the support of framework Django. The frontend is written in HTML, CSS, and JavaScript. The application was tested in lower secondary school. Based on the testing, an access to the files was restricted, now only one user can write new data into the files, which was not considered before. The approach on serving the files was also changed. If the data has been downloaded recently, this data is shown instead of downloading a new set of data. The contribution of my thesis is to ease access to data and to speed up preparations for lessons. During my work, I did not notice any particular limitations of Python that would have prevented me from completing my work.

Keywords

Application, Python, Django, data, mass data processing, asset for teaching

Poděkování

Děkuji vedoucímu práce Mgr. Václavu Šimandlovi, Ph. D. za cenné rady a konzultace při vypracování mé práce a za poskytnutí připraveného školního serveru, na který jsem mohl aplikaci nasadit.

Obsah

1 Úvod

Toto téma jsem si zvolil, abych si prohloubil své znalosti programování v Pythonu a práci s webovými stránkami. Zaujala mě možnost prozkoumat spojení Pythonu a webových stránek včetně hledání limitů tohoto propojení. Záměrem předložené práce je vytvořit aplikaci, která se bude používat v rámci výuky a usnadní přípravu učitelům na výuku. Pedagogové tak budou mít k dispozici aktuální data okamžitě použitelná při výkladu probírané látky.

1.1 Cíle

Cílem praktické části je vytvoření webové aplikace, která bude nabízet aktuální data vhodná pro výuku hromadného zpracování dat. Dalším cílem je pak výběr vhodného frameworku, který poslouží k provedení práce. Při výběru je možné zjistit rozmanitost frameworků na podporu Pythonu, které lze použít a které nikoliv. Dalším cílem je vyhledání vhodných webových stránek se sportovními statistikami, vývojem cen různých komodit na světových trzích a aktuálního počasí v České republice. Z těchto zdrojů budou získávána data a poté nabízena uživateli na webové stránky. Uživatel si na webových stránkách zvolí, jaký typ dat chce a bude si je moct stáhnout. Před stažením se uživateli zobrazí náhled vybraných dat. Stažený soubor bude dostupný ve formátu XLSX nebo CSV. Hlavním přínosem aplikace bude jednodušší prostředí na výběr dat, která se dále budou využívat ve výuce zpracování hromadných dat. Dalším přínosem bude zjištění možností a limitů Pythonu při tvorbě webových aplikací.

1.2 Metody

První aplikovanou metodou mé práce byla analýza. V první řadě nalezení vhodného frameworku pro použití. Rozhodoval jsem se mezi dvěma zaběhlými frameworky a jednou novinkou. Vytvořil jsem si tři testovací aplikace za pomoci každého z daných frameworků a vyzkoušel jejich funkčnost. Poté následovala analýza vhodných webových stránek, ze kterých se čerpají data. Stránky jsem vybíral dle možností stáhnutí dat, jejich celistvosti a také jednoduchosti URL adres na procházení více stránek na jednou. Hlavní metoda práce pak byla tvorba aplikace, při které byl použit vodopádový model vývoje. Dále pak nalezení vhodné metody pro získání dat z vybraných webových stránek a poté celková implementace, aby tyto části spolu fungovaly bez problémů. Backend aplikace byl realizován v jazyce Python, konkrétně verzi 3.11., za použití vybraného frameworku. Frontend aplikace byl napsaná v jazyce HTML za použití kaskádových stylů a Javascriptu. V průběhu psaní byla aplikace nasazená na testovací server společnosti Linode, poté jsem dostal k dispozici připravený školní server, na který jsem aplikaci

nasadil. Po dokončení aplikace jsem vytvořil sadu testovacích úloh a aplikaci otestoval na základní škole. Po prvním testování bylo potřeba aplikaci upravit, ošetřit proti problémům nalezeným během testování. Druhé testování se stejnou sadou úloh proběhla bez problémů.

2 Výběr frameworku

Základním kamenem mojí aplikace je framework, který bude spojovat Python a webové rozhraní. Bylo potřeba vybrat správný framework a otestovat ho na jednoduché aplikaci se vstupem uživatele, aby se zaručila základní funkčnost. Jako testovací aplikaci jsem si vytvořil kód, který simuluje hod třemi kostkami tolikrát, kolikrát zvolí uživatel. Zde popíšu nalezené frameworky, popis testování je v kapitole 5.4.

2.1 Py-script

První framework, který jsem si vybral, je framework Py-script. Jedná se o novinku, která v době začátku psaní mé aplikace nebyla stará ani šest měsíců. Py-script umožnuje psát Python kódy přímo do HTML souborů webové aplikace. Výhoda je ta, že aplikace nebude závislá na externím serveru, kde poběží, ale bude se vykreslovat stejně jako HTML. Tudíž se jedná o framework, který běží na klientovi. Další výhodou je, že není potřeba robustní systém souborů a na jednoduché věci stačí pouze HTML soubor [1].

2.2 Flask

Další framework, na který jsem narazil, je framework Flask, který běží na serveru, nikoliv na klientu jako Py-script. Flask je již zaběhlý framework, v provozu od roku 2010. Je to lehký framework určený pro jednodušší aplikace. Flask umožňuje použití více databází, což je pro můj projekt nepodstatné. Flask nemá žádné šablony, a proto je potřeba si strukturu souborů vytvořit od začátku samostatně [2].

2.3 Django

Poslední framework, který jsem vyhledal je framework Django, který stejně jako Flask, běží na serveru. Je nejstarším ze zmíněných frameworků, funguje od roku 2005 a je robustnější, proto má také složitější souborovou strukturu. Django se nejvíce hodí na vytváření komplexních webových aplikací. Podporuje také více aplikací v rámci jednoho projektu a disponuje předvytvořenými šablonami pro projekty [3].

2.4 Zhodnocení frameworků

Faktem je, že novinka Py-script ještě nebyla pořádně otestovaná a neumí toho tolik jako ostatní dva zmíněné frameworky. Vzhledem k tomu, že při vytváření testovací aplikace vzniklo tolik problémů, které ještě nebyly vyřešeny nebo jejich řešení byla velmi složitá, jsem usoudil,

že by další pokračování tímto směrem nebylo efektivní. Chtěl jsem se vyhnout problémům, které bych nedokázal eliminovat, a které by mi znemožnily dokončit vlastní práci. Navíc je framework stále ve vývoji a rozdíl mezi začátkem a koncem psaní aplikace by mohl být několik nových verzí. Zkušební aplikace již teď nefunguje tak, jak by měla.

Na výběr mi tedy zbývají Django nebo Flask. Věřím, že oba frameworky by byly schopné mou aplikaci v pořádku zvládnout. Oba frameworky nabízí systém dědění HTML souborů z jednoho základního. Flask je oproti Djangu jednodušší v počáteční části tvorby aplikace, což se může jevit jako výhodnější. Nakonec jsem si vybral Django i přesto, že je práce v prví fázi složitější, a to z toho důvodu, že je využívají známé aplikace jako například Instagram, Spotify nebo Youtube [4].

3 Framework Django

3.1 Struktura souborů

Pro pochopení frameworku Djanga je potřeba porozumět struktuře jeho souborů. Po vytvoření nového projektu, pomocí příkazu **django-admin startproject mysite** se vytvoří následující souborová struktura, viz Obrázek 1. Projekt se jmenuje **mysite**.

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

Obrázek 1: Struktura souborů projektu mysite [5]

Soubor manage.py se používá pro práci v příkazovém řádku, například pro spuštění testovacího lokálního serveru. Soubor urls.py propojuje všechny aplikace v projektu a poté rozhoduje podle napsané URL adresy. Soubor settings.py slouží pro nastavení celého projektu. Soubory *asgi.py* a *wsgi.py* jsou připraveny pro komunikací s webovým serverem.

V tomto adresáři pak lze vytvořit několik aplikací, které poběží současně. Příkazem **python manage.py startapp polls** se vytvoří aplikace s názvem **polls**, její struktura je vidět na Obrázek 2 [5].

```
polls/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    urls.py
    views.py
```

Obrázek 2: Struktura souborů aplikace polls [5]

3.2 Základní soubory

Pro základní fungování všech aplikací je potřeba vědět hlavně o dvou souborech. Soubor *urls.py*, *views.py*.

Soubor *urls.py*, který je prakticky seznam všech URL adres, se kterými aplikace pracuje, viz Zdrojový kód 1. V prvním řádku se importuje cesta z hlavního souboru *urls.py*, ve druhém se pak propojí se souborem *views.py* mojí aplikace. Poté následuje pole s cestami na různé zobrazované templaty. To, co se napíše do adresového řádku determinuje, jaká cesta se použije. V adresovém řádku lze použít i takzvanou dynamickou adresu. Jde o případ, kdy se do adresového řádku píše hodnota, jakožto proměnná. Pokud cesta v adresovém řádku odpovídá nějakému záznamu v poli urlpatterns, pak se v souboru views.py zavolá patřičná funkce [6].

```
from django.urls import path
from . import views

urlpatterns = [
    path("articles/2003/", views.special_case_2003),
    path("articles/<int:year>/", views.year_archive),
    path("articles/<int:year>/<int:month>/", views.month_archive),
]
```

Zdrojový kód 1: Ukázka souboru urls.py [6]

V souboru *views.py* se nachází funkce pro každou URL adresu. Po napsání URL adresy do adresového řádku, se zavolá odpovídající funkce ze souboru *views.py*. V těchto funkcích se pak provádí veškeré backendové kódy. Funkce může vrátit přímo napsanou odpověď HTML kódu [7].

Při rozsáhlejší aplikaci ale není vhodné psát HTML stránku do jednoho řádku v souboru *views.py*. Proto je zde druhá možnost, a to zobrazování jiných souborů, takzvaných templatů [8].

Tyto soubory se nachází všechny společně ve složce templates. V souborech se nachází klasický HTML kód pro zobrazení webové stránky [9].

Pokud chceme pracovat s jakýmkoliv jiným souborem, je potřeba ve složce projektu vytvořit složku static, v této složce se uchovávají veškeré ostatní soubory, které nesouvisí s Djangem. Ve složce je pak potřeba vytvořit podsložku vždy pro každou aplikaci. Do této složky se vkládají tyto soubory, kterým Django říká statické, například CSS soubory. [10]

3.3 Systém dědění

Django disponuje systémem dědění pro zobrazované templaty aplikace. Umožnuje tedy vytvořit kostru aplikace v jednom mateřském HTML souboru. Kostra se rozdělí do takzvaných bloků. Synovské soubory pak mohou tyto bloky v případě potřeby přepisovat nebo je zobrazit stejně. Bloky začínají tagem {% block název %} a končí tagem {% endblock %} [9].

Ukázka mateřského souboru ve Zdrojový kód 2.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="style.css">
    <title>{% block title %}My amazing site{% endblock %}</title>
</head>

<body>
    <div id="sidebar">
        {% block sidebar %}
        <ul>
            <li><a href="/">Home</a></li>
            <li><a href="/blog/">Blog</a></li>
        </ul>
        {% endblock %}
    </div>

    <div id="content">
        {% block content %}{% endblock %}
    </div>
</body>
</html>
```

Zdrojový kód 2: Ukázka mateřského souboru [9]

V synovských souborech se pak musí definovat, z kterého mateřského souboru dědí. O to se stará první tag **extends** [9], viz Zdrojový kód 3.

```
{% extends "base.html" %}

{% block title %}My amazing blog{% endblock %}

{% block content %}
{% for entry in blog_entries %}
    <h2>{{ entry.title }}</h2>
    <p>{{ entry.body }}</p>
{% endfor %}
{% endblock %}
```

Zdrojový kód 3: Ukázka synovského souboru [9]

3.4 Formuláře

Django nabízí zjednodušený a ucelený pohled na formuláře. Všechny formuláře se tvoří jako vlastní třída v určeném souboru *forms.py*. V něm se píšou veškeré inputy, které chceme, aby formulář obsahoval, ve Zdrojový kód 4 lze vidět jeden input textového pole [11].

```
from django import forms

class NameForm(forms.Form):
    your_name = forms.CharField(label="Your name", max_length=100)
```

Zdrojový kód 4: Ukázka souboru forms.py [11]

Když se pracuje s formuláři, používají se pouze dvě HTTP metody, GET a POST. Metoda GET posílá nešifrovaná data jako řetězec v URL adrese. Oproti tomu POST posílaná data zašifruje a pošle v balíčku. POST je tedy vhodnější metoda pro posílání například přihlašovacích údajů [11].

V souboru *views.py*, ve Zdrojový kód 5, se pracuje s formulářem podle metody přístupu. Pokud se stránka načítá metodou GET, ukáže se uživateli prázdný formulář k vyplnění. Pokud stránka odesílá data metodou POST, data se zpracují a pošlou na další stránku, dle potřeby.

```
from django.http import HttpResponseRedirect
from django.shortcuts import render

from .forms import NameForm


def get_name(request):
    # if this is a POST request we need to process the form data
    if request.method == "POST":
        # create a form instance and populate it with data from the
    request:
        form = NameForm(request.POST)
        # check whether it's valid:
        if form.is_valid():
            # process the data in form.cleaned_data as required
            # ...
            # redirect to a new URL:
            return HttpResponseRedirect("/thanks/")

    # if a GET (or any other method) we'll create a blank form
    else:
        form = NameForm()

    return render(request, "name.html", {"form": form})
```

Zdrojový kód 5: Ukázka souboru views.py [11]

V HTML souboru se pak místo klasického vypisování celého formuláře vloží tag poslaného formuláře a potvrzovací tlačítko. Dále je zde potřeba vložit tag pro CRSF token. Django takto brání formuláře proti Cross Site Request Forgery, útoku, kdy se jiná stránka tváří jako uživatel a rovněž používá i jeho údaje. Pokud je útok úspěšný, škodlivá stránka pak může vykonávat

nevhodné editování či mazání obsahu, a přitom se tvářit jako legitimně přihlášený uživatel [11],[12].

```
<form action="/your-name/" method="post">
    {% csrf_token %}
    {{ form }}
    <input type="submit" value="Submit">
</form>
```

Zdrojový kód 6: Ukázka soubrou name.html [11]

4 Model vývoje

Při tvorbě aplikace jsem použil tzv. Vodopádový model. Vodopádový model je vývojový proces projektu, kde všechny kapitoly jsou na sebe sekvenčně navázané a není možné pokračovat do další fáze, bez toho, aniž by byla dokončena fáze předchozí. Toto připomíná průtok vody vodopádem, z čehož plyne název tohoto modelu. Model prvně pojmenoval doktor Winston W. Royce [13].

Části modelu podle Royce jsou:

- Požadavky na systém
- Požadavky na software
- Analýza požadavků
- Návrh programu
- Implementace
- Testování
- Provoz

Ve fázi požadavků, ať už na systém nebo na software samotný, se sbírají požadavky na software. Co by měl software umět, pro koho by měl software být, jak by měl fungovat apod. Poté je potřeba požadavky analyzovat, jsou-li vůbec reálné a proveditelné. Pokud existuje více možností provedení požadavku, bude vybírán ten nejlepší způsob, který vyhovuje všem účastníkům vývoje. V návrhu už jsou známy metody, jakými bude software vytvářen a jsou zde uvedeny konkrétní návrhy na aplikaci například grafický design softwaru. Fáze implementace je psaní kódu pro všechny návrhy a požadavky, které aplikace má. Testování aplikace má za úkol odhalit jakékoliv chyby aplikace, ale také ověřit, jestli jsou splněny všechny požadavky. Fáze provozu je finální fáze vývoje, kdy je software spuštěn a už na něm není potřeba pracovat, řeší se pouze údržba softwaru. [13]

Mé části vodopádového modelu jsou:

1. Analýza požadavků na aplikaci
2. Návrh systému aplikace
3. Implementace
4. Nasazení
5. Testování

Oproti základnímu modelu jsem spojil všechny kategorie požadavků do jedné kapitoly. Z fáze implementace jsem vytřídil nasazení na server, které si zaslouží vlastní kapitolu. Jako poslední jsem si zvolil fázi testování.

5 Analýza požadavků na aplikaci

V první řadě byla potřeba určit požadavky, které budeme po aplikaci chtít. V této fázi se určují požadavky od zadavatele projektu. Zadavatele projektu nemám, proto zde vypíšu požadavky, které jsem si stanovil po dohodě s vedoucím práce.

Aplikace poběží na vybraném frameworku, kde hlavní programovací jazyk bude Python. Zobrazení webové stránky bude probíhat zejména s pomocí HTML a CSS, s případným použitím Javascriptu.

5.1 Základní myšlenka

Hlavní myšlenka aplikace je usnadnění práce učiteli informatiky při výuce hromadného zpracování dat. Aplikace bude na jednom místě shromažďovat několik typů těchto dat, ze kterých si učitel vybere ta, která zrovna k výuce potřebuje a nemusí složitě vymýšlet či hledat nová. Data budou připravena pro použití v tabulkových procesorech například k filtrování dat v tabulce, řazení dat v tabulce podle různých kritérií, podmíněnému formátování, výuce grafů nebo funkcí v tabulkových procesorech. Kreativitě se meze nekladou, proto je pouze na učiteli, jakým způsobem získaná data využije.

5.2 Druhy poskytovaných dat

Představoval bych si, aby aplikace měla několik kategorií dat, pro různá použití. První kategorie dat budou tabulky nejvyšších sportovních soutěží v České republice. Druhá kategorie bude aktuální počasí v celé České republice, ale i přehledně rozdělené do menších částí, nejspíše krajů. Poslední kategorií bude cenový vývoj obchodovatelných komodit jako je například zlato, ropa či plyn. Vývoj ceny bude evidován v dostatečném počtu sledovaných období a bude chronologicky seřazen. Je důležité mít sesbírané tématicky různé typy dat tak, aby práce s nimi zaujala co nejširší skupiny žáků.

Všechny kategorie dat budou dostupné ke stažení ve formátu XLSX pro přímou práci s Excelem nebo ve formátu CSV pro práci v jiném tabulkovém editoru jako je například LibreOffice.

5.3 Cílová skupina

Tato aplikace je určena pro výuku informatiky na druhém stupni základní školy a na gymnáziích. Bude sloužit zejména učitelům, ke stažení potřebných dat pro výuku. Pokud učitel ne-

může nebo nechce data předat žákům ve formě staženého souboru, mohou si žáci soubor stáhnout i individuálně a následně začít plnit učitelem stanovené úkoly. Aplikace bude nejvíce pokrývat tematické celky Práce s daty a Hromadná data [14].

Aplikace by měla být dostupná zdarma, bez nutnosti registrace nebo přihlašování. Dále by měla být uživatelsky přátelská pro případ, že učitel zvolí metodu, aby si žáci dané tabulky stahovali sami. Stránka by měla dát na výběr prvně kategorii dat a pak další možné tabulky. Design stránky by měl být minimální, aby děti nerozptyloval při výuce.

5.4 Testování dostupných frameworků

V této části zhodnotím dostupné frameworky, tak, že si z jejich pomocí vytvořím testovací aplikaci. Aplikace bude simulovat hod třemi kostkami tolikrát, kolik zadá uživatel.

5.4.1 Py-script

Ve Zdrojový kód 7 jsou úryvky z kódu testovací aplikace. V hlavičce kódu musí být implementace frameworku, dva řádky pro aktivování Py-scriptu. Následuje formulář pro interakci s uživatelem. Pod tagem *<py-button>* je daný python kód, který se provede po stisknutí tlačítka. První *for cyklus* provede hody kostkou podle vstupu od uživatele, druhý *for cyklus* zapíše výsledný počet do elementů s ID čísla. Pod vstupem se nachází tabulka s buňkami pro dané čísla, kam se výsledná data zapíšou a následně zobrazí.

```
<head>
…
<link rel="stylesheet" href="https://pyscript.net/latest/pyscript.css" />
<script defer src="https://pyscript.net/latest/pyscript.js"></script>
…
</head>
…
<label for="vstup">Zadej počet hodů kostkou.</label>
<input type="number" id="vstup" name="vstup">
<py-button id="click_btn" label="Potvrdit">
  import random
  def on_click(evt):
    soucty = [0]*19
    pocet=int(document.getElementById('vstup').value)

    for i in range(0, pocet):
      soucet = random.randint(1,6)+random.randint(1,6)
+random.randint(1,6)
      soucty[soucet]+=1

    for i in range(0, len(soucty)):
      Element(f"cislo{i}").write (f"{soucty[i]}")
</py-button>
<table>
  <tr> <th>Číslo</th> <th>Kolikrát padlo</th></tr>
  <tr> <td>3</td> <td id = "cislo3"></td> </tr>
  <tr> <td>4</td> <td id = "cislo4"></td> </tr>
  <tr> <td>5</td> <td id = "cislo5"></td> </tr>
  …
  <tr> <td>16</td> <td id = "cislo16"></td> </tr>
  <tr> <td>17</td> <td id = "cislo17"></td> </tr>
  <tr> <td>18</td> <td id = "cislo18"></td> </tr>
</table>
```

Zdrojový kód 7: Testovací kód pro framework py-script

5.4.2 Flask

Z důvodu instalace frameworku Flask, je doporučeno si prvně spustit virtuální prostředí pythonu. V něm je potřeba si naistalovat Flask pomocí *pip install flask*. Flask nenabízí před-připravenou strukturu souborů, proto je potřeba si jí vytvořit ručně. Ve vybrané složce je nutné vytvořit hlavní python soubor, pojmenovaný test.py, a také složku s HTML soubory, pojmenovanou *templates*. Ve Zdrojový kód 8 lze vidět importování frameworku Flask a kódy na provoz frameworku. Stránku, kterou si chceme zobrazit, si musíme prvně definovat jako funkci. Nad danou funkcí se pomocí *@app.route* určí URL stránky, zde prázdné lomítko pro všechny připojení. Ve funkci stránky se pak provádí hlavní kód. Pokud se na stránku přistupuje pomocí metody GET, tak se uživateli zobrazí stránka *index.html* s formulářem.

```
from flask import Flask, render_template, request
import random
app = Flask(__name__)

@app.route("/", methods=["POST", "GET"])
def home():
  if request.method == "POST":
    cislo = request.form["cislo"]
    soucty = [0]*19

    for i in range(0, int(cislo)):
      soucet = random.randint(1,6)+random.randint(1,6)
+random.randint(1,6)
      soucty[soucet]+=1

    return render_template("vysledek.html", soucty=soucty)
  else:
    return render_template("index.html")

if __name__ == "__main__":
  app.run(debug=True)
```

Zdrojový kód 8: Ukázka kódu ze souboru test.py

Formulář se ptá uživatele na počet pokusů a po odeslání znovu načte stránku, jak je vidět ve Zdrojový kód 9. Pokud se na stránku přistupuje metodou POST, to znamená po odeslání formuláře, tak se provede kód házení kostkou a uživateli se zobrazí stránka *výsledek.html*. Společně se zobrazením se na stránku pošle pole s výsledky, ze kterého čerpá výsledná tabulka.

```
<form action="#" method="post">
    <p>Kolikrát hodit?</p>
    <p><input type="number" name="cislo"></p>
    <p><input type="submit" value="submit"></p>
</form>
```

Zdrojový kód 9: Ukázka formuláře ze souboru index.html

Na výsledné stránce se zobrazuje tabulka a data se plní z poslaného pole, jak je vidět na úryvku tabulky ve Zdrojový kód 10.

```
<table>
<tr><th>Číslo</th><th>Kolikrát</th></tr>
<tr><td>4</td><td>{{soucty.3}}</td></tr>
<tr><td>5</td><td>{{soucty.4}}</td></tr>
<tr><td>5</td><td>{{soucty.5}}</td></tr>
…
<tr><td>16</td><td>{{soucty.15}}</td></tr>
<tr><td>17</td><td>{{soucty.16}}</td></tr>
<tr><td>18</td><td>{{soucty.17}}</td></tr>
</table>
```

Zdrojový kód 10: Ukázka tabulky ze souboru vysledek.html

5.4.3 Django

Stejně jako ve Flasku, je vhodné si pro tvorbu aplikace vytvořit virtuální prostředí, aby se nám nemíchaly různé balíčky pythonu. Je potřeba nainstalovat Django pomocí *pip install django* a v daném adresáři poté založit nový projekt pomocí *django-admin startproject Test*. Tohle vygeneruje strukturu souborů potřebnou k ovládání projektu. Dále je potřeba si v projektu vytvořit aplikaci. Díky příkazu *python manage.py startapp kostka* se vytvoří soubory pro ovládání aplikace s názvem kostka. Jméno aplikace je potřeba dopsat do hlavního *settings.py* souboru. V souboru *views.py* je potřeba vytvořit kód pro zobrazování naší požadované stránky. Princip je podobný jako ve frameworku Flask, definujeme funkci stránky, která bude vracet načtení souboru HTML. V adresáři aplikace je potřeba vytvořit soubor *urls.py*, který má na starost směrování podle URL adresy. Ve Zdrojový kód 11 je ukázka nastavení URL domovské adresy, stejně jako připravení souboru v prvních řádcích.

```
from django.urls import path
from . import views

urlpatterns = [
    path("", views.home, name="home")
]
```

Zdrojový kód 11: Ukázka souboru kostka/urls.py

Jelikož Django umožnuje souběh více aplikací, je potřeba nastavit směrování i v souboru *urls.py* v adresáři pro celý projekt. Do předpřipravené struktury stačí pouze doplnit odkaz na soubor *urls.py* aplikace Kostka, viz Zdrojový kód 12.

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include("kostka.urls"))
]
```

Zdrojový kód 12: Ukázka ze souboru Test/urls.py

Django má všechny formuláře na jednom místě. V adresáři naší aplikace si vytvoříme soubor *forms.py*. V něm, vidíte ve Zdrojový kód 13, se vytvoří nová třída pro daný formulář. Do formuláře stačí napsat proměnnou a nastavit jí typ inputu, v našem případě číselné pole. Název proměnné se poté používá při práci se získanými daty.

```
from django import forms

class Hod_kostkou(forms.Form):
  pocet = forms.IntegerField(label="Zadej počet hodů", min_value=1)
```

Zdrojový kód 13: Soubor forms.py

Ve Zdrojový kód 14 můžeme vidět vytvoření kódu, který se provede, pokud uživatel vstoupí na adresu domovské stránky. Důležité je importovat námi vytvořený formulář, který pošleme do HTML souboru při zobrazování. Je zde stejný způsob rozdělení přístupu na stránku pomocí GET anebo POST, podobně jako u Flasku. Podobně se také řeší zobrazení buďto domovské nebo výsledné stránky. Společně se zobrazení HTML souboru se posílají i data, v případě domovské stránky to je formulář k zobrazení a v případě výsledné to je pole s výsledky hodů.

```
from django.shortcuts import render, HttpResponseRedirect
from .forms import Hod_kostkou
import random

def home(response):
  if response.method == "POST":
    form = Hod_kostkou(response.POST)
    if form.is_valid():
      pocet = form.cleaned_data["pocet"]
      soucty = [0]*19
      for i in range(0, pocet):
        soucet = random.randint(1,6)+random.randint(1,6)+
    random.randint(1,6)
        soucty[soucet]+=1

    return render(response,'vysledek.html', {"soucty":soucty})
  else:
    form = Hod_kostkou()
  return render(response,'home.html', {"form":form})
```

Zdrojový kód 14: Soubor views.py

Nyní už můžeme připravit HTML soubory. V adresáři aplikace vytvoříme složku templates, ve které vytvoříme dva soubory. Soubor *home.html* pro domovskou obrazovku s formulářem a soubor *vysledek.html* pro výslednou tabulku. V souboru *home.html* vytvoříme formulář, kde z klasického HTML kódu stačí napsat tag form a do něj vložit potvrzovací tlačítko, jak je vidět ve Zdrojový kód 15. Akce formuláře jsou prázdné uvozovky, to znamená, že se stránka načte znovu. Dále je do formuláře potřeba vložit ochranný kód Djanga proti CRSF útokům a také formulář, který posíláme ze souboru *views.py*.

```
…
Kolikrát chceš hodit kostkami?<br>
<form action="" method="post">
    {% csrf_token %}
    {{form.as_p}}
    <button type="submit" name="Vypocitat" value="Vypocitat">Vypočítat
    </button>
</form>
…
```

Zdrojový kód 15: Úryvek ze souboru home.html

Na výsledné stránce se pak zobrazují data do tabulky, podobně jako u frameworku Flask. Ve Zdrojový kód 16 lze vidět přiřazování daných dat.

```
…
<table>
<tr><th>Číslo</th><th>Kolikrát</th></tr>
<tr><td>3</td><td>{{soucty.3}}</td></tr>
<tr><td>4</td><td>{{soucty.4}}</td></tr>
<tr><td>5</td><td>{{soucty.5}}</td></tr>
…
<tr><td>16</td><td>{{soucty.16}}</td></tr>
<tr><td>17</td><td>{{soucty.17}}</td></tr>
<tr><td>18</td><td>{{soucty.18}}</td></tr>
</table>
…
```

Zdrojový kód 16: Úryvek ze souboru vysledek.html

5.5 Analýza podobných aplikací

Podobnou aplikaci, kterou jsem nalezl, je aplikace mého vedoucího práce pana doktora Šimandla. Nachází se na webové adrese http://simandl.asp2.cz/. Tato aplikace mi na první pohled přijde nepřehledná a nabízí velké množství dat, ve kterém není úplně lehké se vyznat. Co se týče počasí, je zde k dispozici pouze vybraných 40 meteorologických stanic, ve kterých není žádný systém.

Data ke stažení jsou nabízena ve čtyřech formátech. Pro staré verze Excelu formát XLS, pro novější verze Excelu XLSX. Formát ODS pro OpenOffice dokumenty a univerzální formát CSV.

5.6 Nalezení vhodných stránek pro data

Dále bylo potřeba nalézt stránky, které budou vhodné pro stahování dat. První typ stránek jsem vyhledával pro sporty. Chtěl jsem sporty týmové, jelikož sportovní týmové tabulky mají dostatek dat, aby se vyplatilo je získávat. Například tenis má pro každý zápas pouze celkový výsledek a výsledky daných setů a v tabulce pořadí má pouze číslo pořadí a počet bodů daného hráče. Nejpopulárnější týmové sporty, z hlediska registrovaných hráčů, v České republice jsou fotbal, florbal, volejbal, hokej a basketbal [15]. Florbal jsem odstranil pro zachování diverzity mezi sporty, jelikož jsou si s hokejem podobné a hokej je populárnější z hlediska divácké atraktivity [16]. Volejbalová tabulka je poměrně strohá, a navíc postrádá vysvětlivky, z tohoto důvodu byl volejbal rovněž vyřazen. K dispozici zůstaly sportovní informace ze tří oblastí: fotbal, hokej a basketbal. Basketbal a hokej navíc krásně doplňují přestávku fotbalové ligy v zimě. Získání tabulek bylo jednoduché, jelikož se tabulky nachází přímo na stránkách soutěže a již

ve formátu tabulky. Dále jsem prostudoval dané stránky a nalezl i pár zajímavých tabulek, jiných než hlavní tabulka soutěže. Pro basketbal to je historická tabulka všech klubů, pro fotbal a hokej jsou tabulky rozděleny na domácí a venkovní utkání.

Do komodit jsem chtěl zahrnout především zlato, ropu a elektřinu. Zlato, protože je to jistý a oblíbený investiční prvek, elektřina je nezbytným produktem pro fungování domácností spotřebitelů, státu a zejména podniků ve všech odvětvích národního hospodářství a nakonec ropa, která se využívá na výrobu benzínu a nafty do automobilů. Ceny ropy a elektřiny významně určují ceny konečných výrobků, a proto je dobré vědět, jaká je jejich cena a jak se v čase vyvíjí.

Nalezení stránek komodity už bylo podstatně složitější. Na většině stránek se nachází jako text pouze aktuální hodnota a historie bývá ukryta pod nějakým grafem. Stránky jako https://zlataky.cz/cena-zlata-a-grafy-zlata a https://goldprice.org/ proto nebylo možné použít. Stránka https://www.kurzy.cz/komodity/zlato-graf-vyvoje-ceny/ vypadá na první pohled stejně jako předchozí dvě, kde se historie uchovává pouze ve formě grafu. Po chvilce hledání jsem však objevil, že stránka nabízí pro každý měsíc své vlastní statistiky. V URL adrese se pak lehce dá vyměnit datum, dokonce i měna a hmotnost, která se zobrazí. Například následující adresa odkazuje na rok 2023, září a ceny se zobrazí v korunách na gram. https://www.kurzy.cz/komodity/zlato-graf-vyvoje-ceny/202309-czk-1g. Tato zaměnitelnost je skvělá pro psaní kódu na stažení celé historie.

Na této stránce se nachází i ropa a elektřina, které jsem chtěl implementovat. Z důvodu dobré zaměnitelnosti pro získávání dat, jsem se rozhodl zůstat na této stránce a přidat ještě zbylé komodity, které se na stránce nacházejí. Z investičních drahých kovů jsem se rozhodl přidat ještě stříbro a platinu. Na stránce se nachází dva typy ropy, proto implementuju oba typy. K elektřině jsem přidal další prvek z domácnosti a to zemní plyn.

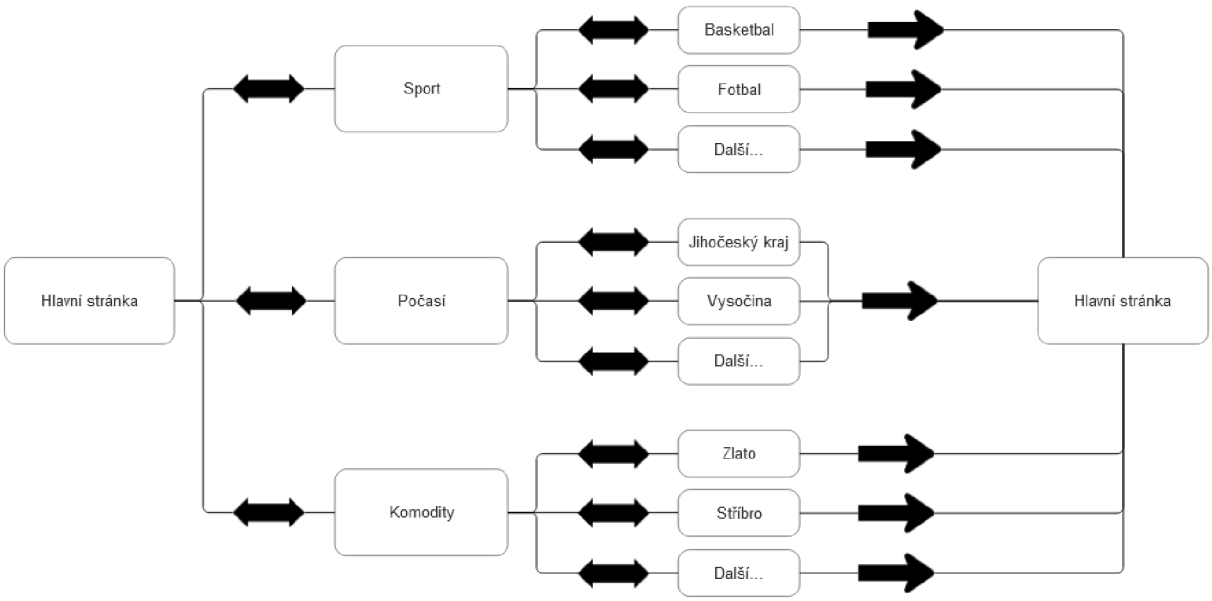
Pro shromažďování dat o počasí byla situace jednoduší, protože na většině stránek se nachází data ve formátu čísel jako text, který půjde lehce stáhnout. https://pocasi.seznam.cz byla první volba. Název města i teplota byla v pořádku, ale na stránce mi chybělo podrobnější rozdělení měst. Nachází se zde pouze abecední seznam měst, což není vhodné k dalšímu využití. Další v pořadí analyzovaných stránek byla https://www.in-pocasi.cz/, kterou jsem vyhodnotil jako vhodnou pro zpracování další části mé bakalářské práce. Česká republika je zde rozdělena do krajů, každý kraj nabízí všechna okresní města a pokud bude potřeba, lze další významná města doplnit ze seznamu zohledňujícího rovněž jednotlivé kraje. Navíc se zde nachází údaj o nadmořské výšce, což se může využít jako další data, například k porovnávání teplot v různých výškách. Vybral jsem proto všechna okresní města pro daný kraj a doplnil je o další zajímavá města, například s velmi rozdílnou nadmořskou výškou.

6 Návrh systému aplikace

Druhá část modelu se věnuje návrhu systému aplikace.

6.1 Použití aplikace

Aplikace bude mít pouze jeden typ použití, nezávislý na typu uživatele. Nezáleží na tom, jestli se bude jednat o učitele nebo o žáky, všichni budou mít stejná práva a možnosti. Uživatel si zobrazí hlavní stránku aplikace, kde bude popis aplikace a formulář s výběrem tří typů dat. Po potvrzení typu dat, bude uživateli zobrazena stránka, kde si bude moct vybrat už specifická data. Například u sportu již budou na výběr tabulky různých sportů, u počasí rozdělení na kraje a podobně. Po zvolení specifické tabulky, bude uživateli zobrazena další stránka, už poslední, kde bude návrh tabulky a možnost jejího stažení do souboru ve dvou různých formátech. Z rozcestníků se bude možné vrátit na hlavní stránku nebo dále pokračovat na specifikace. Ze specifických tabulek bude umožněno vrátit se na rozcestníky anebo až na hlavní stránku, viz Obrázek 3.



Obrázek 3: Použití aplikace

6.2 Seznam všech HTML souborů

Celkem bude potřeba vytvořit 34 HTML souborů. Jeden soubor bude základní HTML soubor, ze kterého pak budou ostatní dědit. Jeden soubor bude obstarávat hlavní stránku, tři soubory budou potřeba pro rozcestníky. Pro tabulkové stránky je to tedy 29 souborů, rozdělené takto do svých skupin.

V počasí se jedná o 14 souborů:

- Všechny kraje dohromady
- Středočeský kraj + Praha
- Jihočeský kraj
- Plzeňský kraj
- Karlovarský kraj
- Ústecký kraj
- Liberecký kraj
- Královehradecký kraj
- Pardubický kraj
- Olomoucký kraj
- Moravskoslezský kraj
- Zlínský kraj
- Jihomoravský kraj
- Kraj Vysočina

V komoditách se jedná o 7 souborů:

- Vývoj ceny zlata
- Vývoj ceny stříbra
- Vývoj ceny platiny
- Vývoj ceny ropy BRENT
- Vývoj ceny ropy WTI
- Vývoj ceny elektřiny
- Vývoj ceny zemního plynu

Ve sportu se jedná o 8 souborů:

- Basketbalová aktuální tabulka
- Basketbalová historická tabulka
- Fotbalová kompletní tabulka
- Fotbalová tabulka domácích utkání
- Fotbalová tabulka venkovních utkání
- Hokejová kompletní tabulka
- Hokejová tabulka domácích utkání
- Hokejová tabulka venkovních utkání

6.3 Rozšiřitelnost aplikace

Aplikace bude strukturovaná tak, aby se dala kdykoliv rozšířit přidáním dalších výběrů. Momentálně jsou na výběr pouze tři typy dat. Do budoucna se však může najít jiný typ dat, který má unikátní hodnoty a bude sloužit lépe k některým výukovým cílům.

6.4 Návrh scrapování

Ke stažení dat z webových stránek použiju metodu zvanou web scraping, česky také extrakce dat z webu. Jedná se o proces sbírání a zpracovávání dat z webových stránek. I když tyto akce mohou být prováděny ručně člověkem, většinou se jedná o různé boty či automatizované aplikace. První fáze scrapingu je získání html kódu. Z webové stránky se získá HTML kód tak, jako kdyby se stránka měla uživateli zobrazit. V druhé fázi se už tento kód dále upravuje a lze použít k zamýšleným potřebám. [17]

6.4.1 Počasí

Každá stránka města vypadá stejně, což je výhoda pro hromadné získávání dat. Na Obrázek 4 je vidět část stránky pro město České Budějovice, která bude potřeba. Zde jsou vidět všechna data, které budeme potřebovat. Název města se nachází v tagu <h1>. Informace o teplotě, pocitové teplotě a vlhkosti jsou v tagu s různou třídou. Informaci o nadmořské výšce města vezmu z odstavce popisu, vykrojení závorek by neměl být problém.

Obrázek 4: Část stránky pro město České Budějovice

Ke staženým datům ještě přidám údaje pro den a pro čas, kdy se daná data stáhla. Tímto umožním srovnávání počasí jak mezi různými městy, tak i ve stejných městech, ale v jiný den, či hodinu. Výsledná tabulka bude vypadat přibližně jako tabulka na Obrázek 5.

Den	Čas	Město	Nadmořská výška	Tepolota	Pocitová teplota	Vlhkost
01.01.2024	12:00	Město 1	N. výška 1	Teplota 1	P. teplota 1	Vlhkost 1
01.01.2024	12:00	Město 2	N. výška 2	Teplota 2	P. teplota 2	Vlhkost 2
01.01.2024	12:00	Město 3	N. výška 3	Teplota 3	P. teplota 3	Vlhkost 3
01.01.2024	12:00	Město 4	N. výška 4	Teplota 4	P. teplota 4	Vlhkost 4

Obrázek 5: Návrh tabulky pro počasí

Města vyberu z mapy měst na serveru https://www.in-pocasi.cz/. Mapa je tam rozdělena po krajích, které si lze rozkliknout a zobrazit výčet okresních měst.

Stahování dat rozdělím na kraje, kde budou uvedena všechna okresní města a k nim přidám ještě další jiná města například s velmi rozdílnou nadmořskou výškou.

Při požadavku uživatele na stažení dat se vyčistí starý soubor určený pro daný kraj. Poté se soubor zaplní novými daty a je nabídnut uživateli ke stažení v obou formátech.

6.4.2 Sport

Oproti počasí a komoditám, máme ve sportu ucelenou tabulku na jedné stránce, kterou je potřeba stáhnout jako celek. Je vhodné, aby prošla lehkými úpravami, to ale budu popisovat až u každé tabulky zvlášť. Jelikož jsou tabulky proměnlivé, je potřeba je stahovat pokaždé znovu. Při každém novém stažení se bude na serveru přepisovat starý soubor novými daty, podobně jako u počasí.

Basketbal

Na Obrázek 6 je vidět tabulka, tak jak se zobrazí na stránce https://nbl.basketball/tabulka. Z výsledné tabulky bude potřeba odstranit logo týmu a posun týmu ze začátku tabulky. Z konce tabulky pak ukazatel posledních pěti zápasů a oranžové plus, které po kliknutí zobrazí graf vývoje pořadí daného týmu.

Dále z tabulky odstraním nadbytečné sloupce úspěšnost, rozdíl, procenta domácích a venkovních výher. Nadbytečné jsou z toho důvodu, že je můžeme vypočítat z předchozích dat. Může se tedy jednat o další úkoly pro žáky.

POŘADÍ		TÝM	Z	V	P	ÚSPĚŠNOST	VSTŘELENO	OBDRŽENO	ROZDÍL	DOMA	VENKU	DV	DP	DV(%)	VV	VP	VV(%)	POSLEDNÍCH 5 ZÁPASŮ	
1		ERA Basketball Nymburk	22	17	5	77.3	2.038	1.552	486	11	11	10	1	90.9%	7	4	63.6%		
2		BK Opava	22	17	5	77.3	2.008	1.818	190	11	11	9	2	81.8%	8	3	72.7%		
3		SLUNETA Ústí nad Labem	22	14	8	63.6	1.983	1.828	155	11	11	7	4	63.6%	7	4	63.6%		
4		BK ARMEX Děčín	22	13	9	59.1	1.905	1.799	106	11	11	8	3	72.7%	5	6	45.5%		
5		Basket Brno	22	12	10	54.5	1.686	1.736	-50	11	11	8	3	72.7%	4	7	36.4%		
6		USK Praha	22	11	11	50.0	1.650	1.678	-28	11	11	8	3	72.7%	3	8	27.3%		

Obrázek 6: Část tabulky basketbalové soutěže

Aby data měla stejný styl, je potřeba ještě udělat několik úprav v hlavičce tabulky. Názvy upravím tak, aby byly srozumitelné pro všechny a nemusela být na stránce legenda. Poté by záhlaví tabulky mělo vypadat tak, jako Obrázek 7.

Pořadí	Tým	Zápasy	Výhry	Prohry	Vstřeleno	Obdrženo	Doma	Venku	Výhry Doma	Prohry Doma	Výhry Venku	Prohry Venku

Obrázek 7: Návrh záhlaví tabulky pro basketbal

Stejným způsobem stáhneme i tabulku historických hodnot. Tabulka má pouze jiné hodnoty, způsob stažení zůstává stejný.

Fotbal

Postup bude podobný jako u basketbalu. Z tabulky, ze stránky https://www.fortunaliga.cz/tabulka, je potřeba odstranit logo týmu, posun v tabulce a koncový graf. Opět odstraníme nadbytečné sloupce, zde pouze rozdíl v gólech. Z tabulky na Obrázek 8 získáme záhlaví tabulky na Obrázek 9. Tabulky pro domácí a venkovní utkání jsou vzhledově úplně stejné, takže vyžadují podobné úpravy.

#		KLUB	Z	V	R	P	G+	G-	RG	B	BK	
1.	–	SPARTA	22	19	2	1	55	15	+40	59	2,682	⊕
2.	–	SLAVIA	22	17	4	1	46	17	+29	55	2,500	⊕
3.	–	PLZEŇ	22	14	3	5	52	26	+26	45	2,046	⊕
4.	–	SLOVÁCKO	23	11	5	7	33	27	+6	38	1,652	⊕
5.	↑	ML. BOLESLAV	23	9	5	9	40	38	+2	32	1,391	⊕
6.	↓	OSTRAVA	22	9	4	9	31	26	+5	31	1,409	⊕

Obrázek 8: Část tabulky fotbalové soutěže

Pořadí	Klub	Zápasy	Výhry	Remízy	Prohry	Góly+	Góly-	Body

Obrázek 9: Návrh záhlaví tabulky pro fotbal

Hokej

I u hokeje je postup obdobný jako u ostatních sportů. Na Obrázek 10 lze vidět tabulku tak, jak je na stránce https://www.hokej.cz/tipsport-extraliga/table. U hokejové tabulky jsou k odstranění z důvodu nadbytečnosti pouze dva záznamy, a to využití přesilovek a ubráněná oslabení. Hokejová tabulka je mnohem větší než tabulka ostatních sportů a pravděpodobně se nevejde na stránku s přepsanou hlavičkou. Toto zjištění znamená, že bude potřeba vytvořit na stránku legendu.

#	TÝM	Z	V	VP	PP	P	SKÓRE	B	B%	SB	SP	PŘ	GPŘ	VPŘ	OS	OGOS	UOS	VGOS	VV	H	T
1	HC Dynamo Pardubice	51	32	8	6	5	196:113	118	77	31.73	25.75	175	47	26.86	161	31	80.75	6	55.39	294	420
2	HC Sparta Praha	51	31	6	3	11	170:118	108	71	32.27	25.75	176	34	19.32	189	28	85.19	10	53.15	270	603
3	HC Oceláři Třinec	51	25	9	5	12	170:129	98	64	28.71	29.24	173	41	23.70	176	32	81.82	3	51.67	287	470
4	HC Kometa Brno	51	21	8	5	17	157:137	84	55	30.84	28.29	182	39	21.43	167	34	79.64	4	47.54	295	499
5	Banes Motor Č. Budějovice	51	23	2	10	16	145:142	83	54	29.51	26.73	162	33	20.37	174	34	80.46	10	51.09	252	557
6	HC VERVA Litvínov	51	23	4	4	20	159:162	81	53	31.24	33.20	207	44	21.26	185	41	77.84	2	42.94	380	506

Obrázek 10: Část tabulky hokejové soutěže

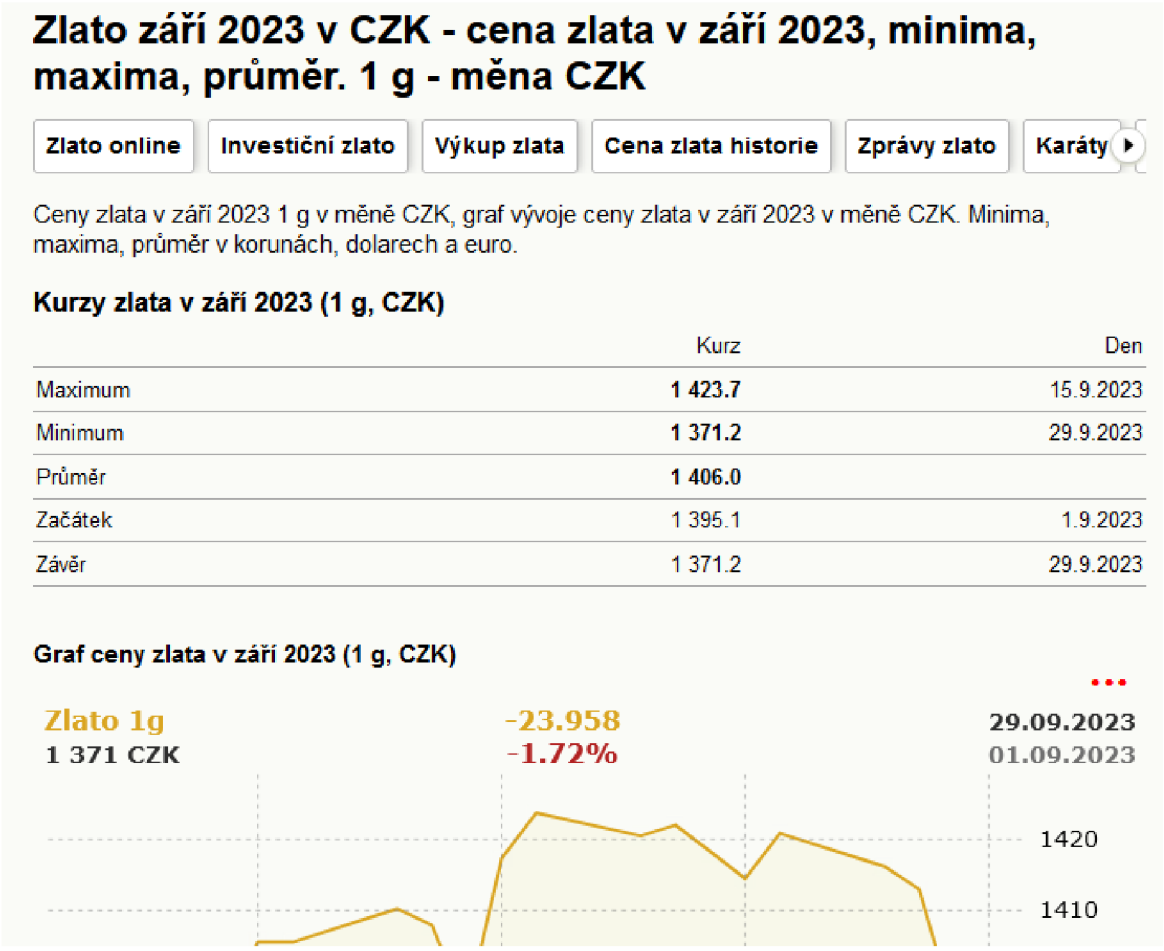
Záhlaví výsledné stažené tabulky by pak měla vypadat jako tabulka z Obrázek 11. Tabulky pro domácí a venkovní utkání vypadají úplně stejně.

#	Tým	Z	V	VP	PP	P	Skóre	B	B[%]	SB	SP	Př	GPř	Os	OGOs	VGOs	VV[%]	H	T

Obrázek 11: Návrh záhlaví tabulky pro hokej

6.4.3 Komodity

Komodity budu stahovat ze stránek https://www.kurzy.cz/komodity/zlato-graf-vyvoje-ceny/202309-czk-1g. V adrese můžeme vidět komoditu, o kterou se jedná a v poslední části za lomítkem lze vidět měsíc, ze kterého data jsou a jednotky, ve kterých se stránka zobrazuje. Na této adrese, ke které patří Obrázek 12, se zobrazuje zlato v září roku 2023. Ceny se zobrazují v korunách za gram. Potřebná data se nachází v tabulce, takže je opět nebude těžké stáhnout.



Obrázek 12: Stránka historie cen zlata

Z této stránky použijeme datum, průměrnou, minimální a maximální cenu a upravíme hlavičku, tak, že do ní připíšeme jednotky. Výsledná tabulka by měla vypadat tak, jako Obrázek 13.

Rok	Měsíc	Průměr[CZK/g]	Maximum[CZK/g]	Minimum[CZK/g]
2008	leden	507,25	524,81	492,26

Obrázek 13: Návrh tabulky pro komodity

Takto se stáhne jeden měsíc. Záměnou čísel v adresovém řádku půjde lehce udělat for cyklus na procházení všech měsíců od daného data. Při každém kliknutí na tlačítko pro stažení, se bude kontrolovat, jestli soubor již existuje nebo ne. V případě, že soubor neexistuje, stáhne se celá historie ceny a zapíše se do nově vytvořeného souboru. V případě, že soubor již existuje, tak se musí zkontrolovat datum posledního zápisu a podle něho dopsat zbylé měsíce do již existujícího souboru.

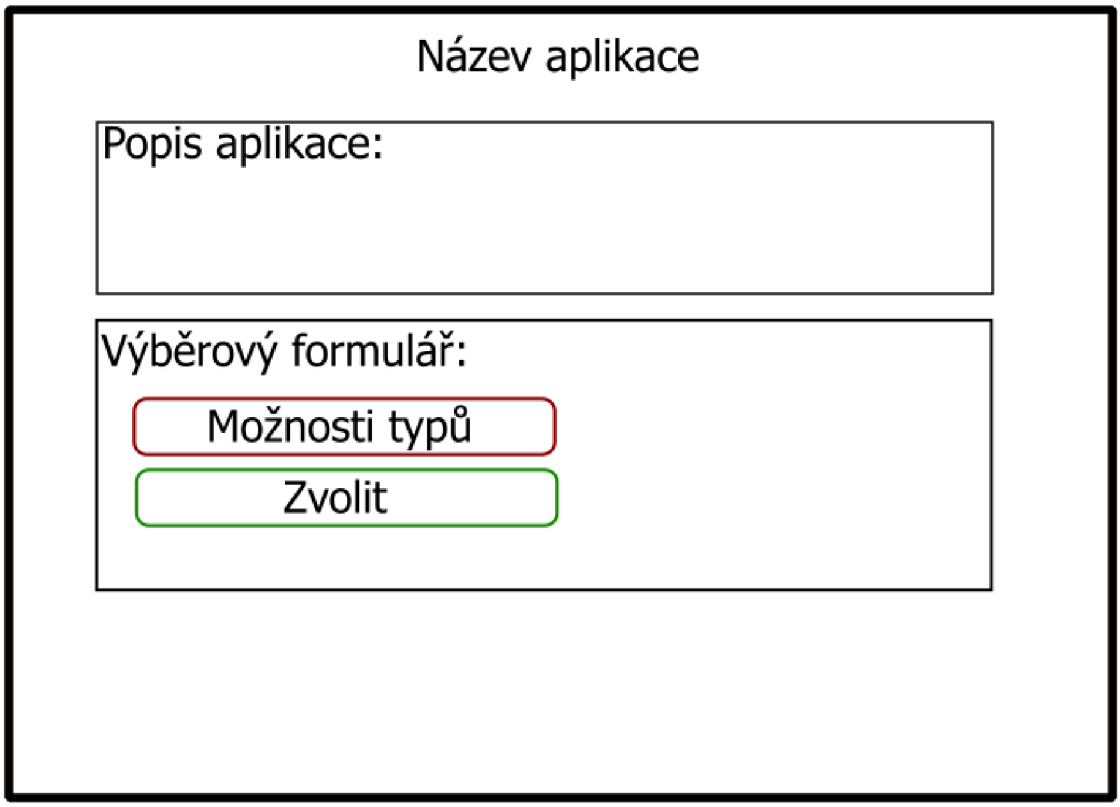
Podobným způsobem stáhnu i data pro ostatní komodity, hlavní rozdíl bude v jednotkách. Pro zlato, stříbro a platinu se jedná o korunu za gram. Pro oba typy ropy se jedná o korunu za litr. Pro elektřinu jde o korunu za megawatthodinu a pro zemní plyn jde o korunu za metr krychlový.

6.5 Grafický návrh

Aplikace by měla mít jednoduchý design. Pokud učitel zvolí metodu, že si žáci budou stahovat data sami, potom na stránce nesmí být nic co by je rozptylovalo. Ze stejného důvodu musí být jasné kam kliknout, abychom se dostali tam, kam potřebujeme. Na všech stránkách bude nahoře hlavička s názvem aplikace, možná s logem. Na všech dalších stránkách pak bude v hlavičce odkaz na stránku předchozí a hlavní. V následujících obrázcích jsou červeně formulářové vstupy typu select, zeleně jsou tlačítka a modře jsou odkazy.

Hlavní stránka

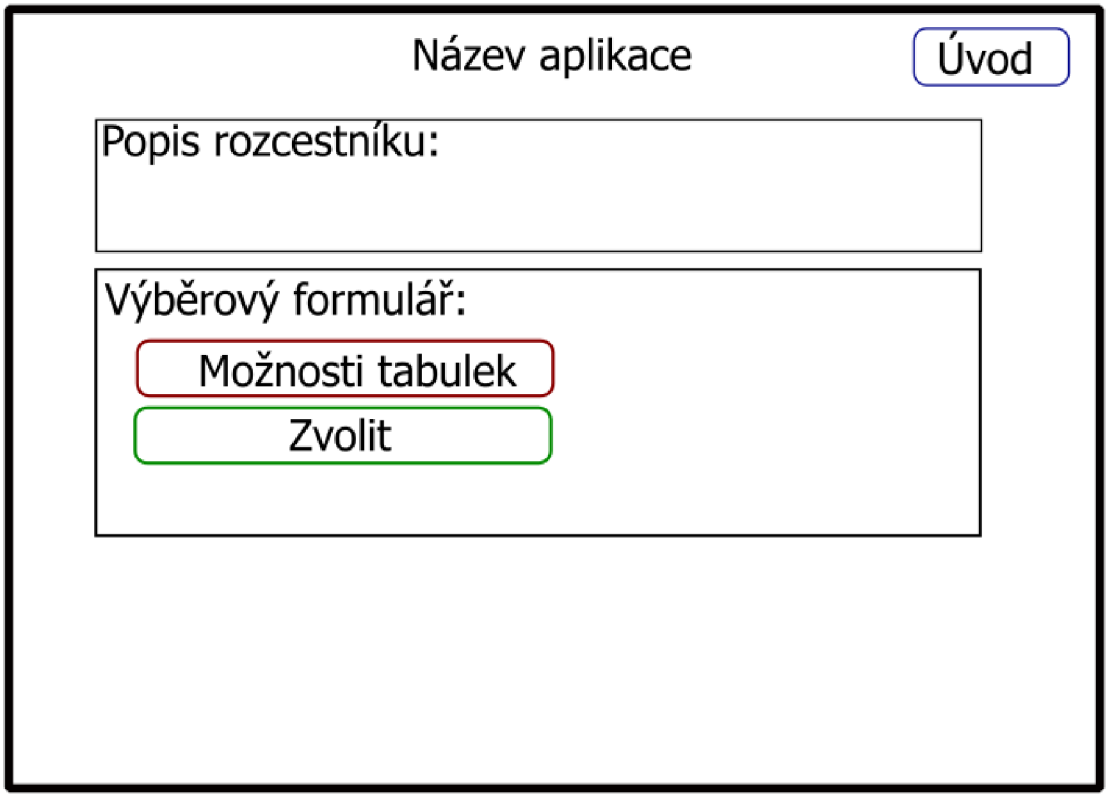
Na hlavní stránce bude nahoře krátký popis aplikace. Pod ním už bude samotný formulář s jedním výběrovým oknem typu select a tlačítkem na potvrzení.



Obrázek 14: Hlavní stránka aplikace

Rozcestníky

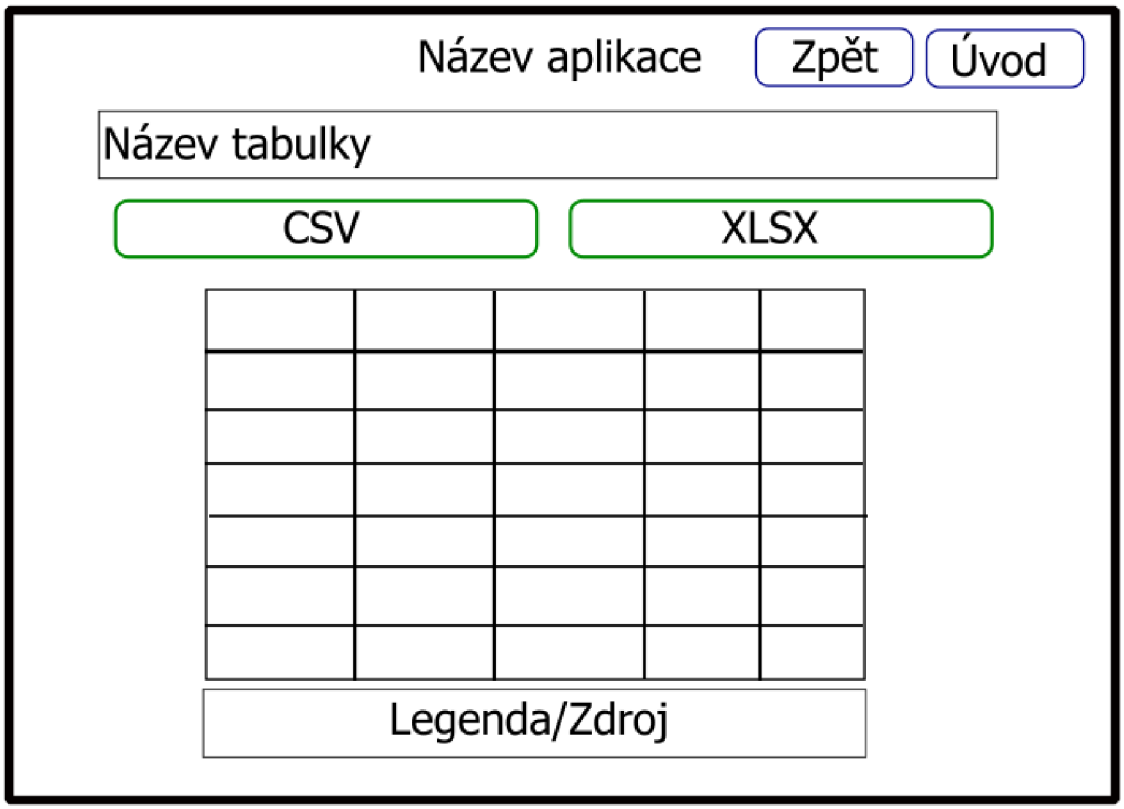
Na stránce rozcestníků bude také krátký popis toho, o čem jsou daná data a vzhledově stejný formulář, který bude nabízet další možnosti rozdělení, opět typu select a tlačítkem na potvrzení. V pravém horním rohu pak bude odkaz zpět na domovskou stránku.



Obrázek 15: Stránka rozcestníku aplikace

Tabulková stránka

Na stránce tabulky pak bude název tabulky, dvě tlačítka na stažení a daná tabulka, aby uživatel viděl data ještě před stažením. Nad tabulkou budou dvě tlačítka na stažení souborů ve formátu CSV a XLSX. Další možnost byla mít tlačítka pod tabulkou, ale u delších tabulek, jako například komodit, by bylo pro uživatele nekomfortní sjíždět pod celou tabulku a až tam hledat tlačítka. Naopak pod tabulkou budou informace, které nejsou tak důležité jako stahovací tlačítka. Na všech stránkách bude uvedena informace odkud jsou data stahována. Na hokejových tabulkách ještě bude připojena legenda vysvětlující zkratky v hlavičce tabulky.



Obrázek 16: Tabulková stránka aplikace

6.6 Soubory ke stažení

Všechny soubory, určené pro uživatele, se budou ukládat na serveru, na kterém aplikace poběží. Uživatel má na výběr ze dvou typů formátů, proto musí být každý soubor na serveru dvakrát, jednou jako formát XLSX a podruhé jako CSV. U souborů pro počasí a sport se jedná vždy o aktuální data, takže se soubory mohou pokaždé nahradit nově staženými daty. Staré soubory se budou celé přepisovat novými daty, aby nedošlo k přehlcení serveru a nemuselo se řešit jejich případné vymazávání. U komodit je historická cena neměnný faktor, a proto nebude potřeba pokaždé stahovat celou historii ceny, ale pouze přidávat nové záznamy, pokud budou k dispozici. Celkový počet souborů určených pro data by tak měl být dvojnásobný počet specifických dat. Jedná se tedy o 29 různých dat ve dvou formátech, to dává dohromady 58 souborů uchovávaných ke stažení.

7 Implementace

Po fázi návrhu pokračujeme fází implementace čili postupem, jak jsem práci tvořil. Uvedu zde několik stěžejních bodů, které jsou důležité či nějak zajímavé.

7.1 Zobrazování stránek za použití Djanga

Jak jsem uvedl v teoretické části 3.2, pro zobrazení jakékoliv stránky pomocí frameworku Django je potřeba několik hlavních souborů. Zde předvedu ukázky daných souborů, pro zobrazování mojí domovské stránky. Jak je vidět ve Zdrojový kód 17, pro domovskou obrazovku jsem nechal prázdné uvozovky, takže zadáním pouze adresy serveru se dostaneme na úvodní stránku aplikace.

```
From django.urls import path
from . import views
urlpatterns = [
      path("", views.home, name="home"),
      path("Pocasi", views.Pocasi, name="Pocasi"),
      path("vsechny", views.vsechny, name="vsechny"),
      …další URLs krajů…
      path("Komodity", views.Komodity, name="Komodity"),
      path("zlato", views.zlato, name = "zlato"),
      …další URLs komodit…
      path("Sport", views.Sport, name="Sport"),
      path("Aknbl", views.Aknbl, name="Aknbl"),
      …další URLs sportů…
    ]
```

Zdrojový kód 17: úryvek ze souboru urls.py

Po zadaní cesty do adresového řádku a zvolení správné cesty z *urls.py* se spustí soubor *views.py*. Soubor, ve kterém se rozhoduje, co se zobrazí. Na domovské stránce se prvně ověří, jestli se posílají nějaká data metodou POST nebo ne. Pokud ne, zobrazí se formulář pro výběr rozcestníků. Pokud ano, tak se stránka přesměruje na vybranou stránku, viz. Zdrojový kód 18.

```
def home(response):
    if response.method == "POST":
      form = Vyber(response.POST)
      if form.is_valid():
        vyber = form.cleaned_data["data"]
        return HttpResponseRedirect(vyber)
    else:
      form = Vyber()

    return render(response, 'main/home.html',{"form":form})
```

Zdrojový kód 18: Zobrazení domovské stránky v souboru views.py

Pokud je potřeba zobrazit formulář, vybere se vytvořený formulář ze souboru *forms.py*. Pro naší domovskou stránku je to tento, níže uvedený Zdrojový kód 19. Label formuláře záměrně neuvádím, otázka na výběr je položena v HTML souboru.

```
class Vyber(forms.Form):
      moznosti = [("Pocasi", "Počasí"),
                  ("Komodity", "Komodity"),
                  ("Sport", "Sport")
                  ]
      data = forms.ChoiceField(choices=moznosti, label="")
```

Zdrojový kód 19: Formulář domovské stránky v souboru forms.py

Posledním souborem potřebným k zobrazení je jakýkoliv HTML template. Zde ukážu soubor *home.html*, který je vidět na Zdrojový kód 20. Na začátku souboru se vyskytuje propojovací kód se souborem *base.html*, který je společným základem všech zobrazovaných stránek. Fungování souboru *base.html* je vysvětleno v teoretické části mé práce. V bloku popis se nachází několik odstavců pro popis aplikace. Blok tabulka na stránce přepisuje kód ze souboru *base.html*, jelikož na hlavní stránce žádnou tabulku vykreslovat nepotřebujeme. V bloku content se pak zobrazuje formulář djanga, princip je také zmíněný v teorii.

```
{% extends "main/base.html" %}
{% block title %}Hromadná data pro výuku{%endblock%}

{%block popis%}
<p>…popis aplikace…</p>
{%endblock%}

{%block tabulka%}
{%endblock%}

{%block stahnout%}
{%endblock%}

{% block content%}
<form action= "", method="post">
{% csrf_token %}
{{form.as_p}}
<button type="submit" value="zvolit" class="zvolit" name =
    "Zvolit">Zvolit</button>
</form>
{% endblock %}
```

Zdrojový kód 20: Ukázka domovské stránky v souboru home.html

7.2 Scrapování Počasí

V hlavním souboru *views.py* je potřeba vytvořit funkci pro rozcestníkovou stránku počasí a také zvlášť pro každý kraj. Na rozcestí se bude zobrazovat formulář nebo nás stránka přesměruje na výslednou tabulku, podle metody přístupu, podobně jako na úvodní stránce. Ve Zdrojový kód 21, lze vidět zobrazení formuláře, který nabízí jednotlivé kraje a podle toho pak zobrazí výslednou stránku.

```
def Pocasi(response):

    if response.method == "POST":
        form = PocasiF(response.POST)
        if form.is_valid():
            vyber = form.cleaned_data["data"]
            return HttpResponseRedirect(vyber)
    else:
        form = PocasiF()

    return render(response, 'main/Pocasi.html', {"form":form})
```

Zdrojový kód 21: Funkce ze souboru views.py pro stránku rozcestí počasí

Každá zobrazovaná stránka, všechny stránky jsou vypsané v kapitole 6.2, musí mít vlastní funkci v souboru *views.py*, vlastní html stránku, která se bude zobrazovat a musí být přidán do souboru *urls.py* kvůli propojení. Jelikož se všechny města stahují stejně, bez rozdílu, tak si

vytvořím vlastní funkci pro stažení dat jednoho města a tu pak budu opakovat pro ostatní. Vytvořená funkce je vidět na Zdrojový kód 22. Funkci jsem si pojmenoval **scrapepocasi** a jako nutné proměnné se zadává cesta souboru, jaký soubor se má otevírat, a pole s adresovými řádky měst. Adresa, ze které se bude scrapovat se skládá ze základu https://www.in-pocasi.cz/predpoved-pocasi/cz/, který je stejný pro všechna města a přidá se k ní zbylá část adresy z pole měst. Dále se otevře soubor podle zadané cesty a napíše se mu hlavička. Ve for cyklu, který projede všechna města ze seznamu, spojím základní adresu s adresou města a vytvořím dotaz pro danou stránku. Dotaz se pošle na stránku a zpátky ze stránky přijde její zdrojový HTML kód. Vrácený HTML kód nemusí být vždy ve správném formátu, proto je upraven do jeho čitelnější podoby. Z tohoto naformátovaného HTML kódu pak vypreparuji potřebné údaje, které zapíšu do otevřeného souboru. Ve for cyklu se nachází testování chyb pomocí příkazu try a except. Tento kód otestuje, jestli se při procházení for cyklu neobjeví nějaká chyba, například, že nepůjde stáhnout jedno město. Pokud chyba nastane, tak kód pokračuje dál, bez chybové hlášky a spadnutí systému. Pouze se dané město vynechá a uživatel nic nepozná.

Problémy

Při stahování dat jsem se setkal s několika problémy. První byl problém s různými časovými pásmy. Ještě když jsem měl aplikaci na testovacím serveru, tak ten, jelikož byl někde v oblasti Londýna, měl jiný čas, než by měl uživatel tady v České republice. Musel jsem tedy importovat knihovnu pytz, která do pythonu přináší časové zóny [18]. Čas se nyní bude brát vždy z časové zóny Evropy čili UTC +1. To způsobí, že jakékoliv stažení dat bude mít evropský čas, i když bude třeba z Ameriky. Mé aplikaci to ale vadit nebude, jelikož je určená pro české školy v našem časovém pásmu.

Druhý menší problém nastal při stahování nadmořské výšky Prahy. Praha jako jediné město nemá uvedenou nadmořskou výšku v závorkách, a proto na ni nefunguje stažení odstavce a vybrání nadmořské výšky z daného odstavce, tak jako u všech ostatních měst. Zkontroloval jsem, zda je Praha opravdu jedinou, která to tak má. Z tohoto důvodu jsem se rozhodl nadmořskou výšku Praze natvrdo napsat jednou podmínkou if, jak je vidět ve Zdrojový kód 22, na konci for cyklu. Data zapisuji odděleně středníkem, bližší důvod je vysvětlen v kapitole 7.7.

```
def scrapepocasi(cestasouboru, mesta):
  base = "https://www.in-pocasi.cz/predpoved-pocasi/cz/"

  with open(cestasouboru, "w", encoding="utf-8") as nove:
        nove.write("Den;Čas;Město;Nadmořská
    výška[m.n.m.];Teplota[°C];Pocitová teplota[°C];Vlhkost[%]\n")

  for i in range(0, len(mesta)):
    try:
      url = base + str(mesta[i])
      dotaz = requests.get(url)
      web = dotaz.text
      soup = BeautifulSoup(web, "html.parser")
      with open(cestasouboru, "a", encoding="utf-8") as file:
        tz = pytz.timezone("Europe/Prague")
        cas_temp = datetime.now(tz)
        cas = cas_temp.strftime("%H:%M")
        den = cas_temp.strftime("%d.%m.%Y")
        mesto = soup.find(name="h1", class_="mb-0").text.strip()
        mesto = mesto.replace("Předpověď počasí", "").strip()
        teplota = soup.find(name="div", class_="alfa mb-
    1").string.strip()
        teplota = teplota.replace(".", ",")
        teplota = teplota.replace("°C", " ")
        zbytek = soup.findAll("span", class_="strong text-black")
        pocit = zbytek[0].string.strip()
        pocit = pocit.replace("°C", "")
        vlhkost = zbytek[1].string.strip()
        vlhkost = vlhkost.replace("%","")
        nadMvyskatemp = soup.find(name= "p").string.strip()
        nadMvyskatemp = nadMvyskatemp.split(")")
        nadMvyskatemp = nadMvyskatemp[0].split("(")
        nadMvyskatemp = nadMvyskatemp[1].split(" ")
        nadMvyska = nadMvyskatemp[0]
        if mesto == "Praha":
          nadMvyska = 280

        data =
f"{den};{cas};{mesto};{nadMvyska};{teplota};{pocit};{vlhkost}" + "\n"
        file.write(data)
    except:
      pass
```

Zdrojový kód 22: vlastní funkce pro scrapování počasí

Pro každý kraj pak stačí vytvořit funkci, viz. Zdrojový kód 23, která vykoná kód při zobrazení stránky daného kraje. Je potřeba do funkce pro stahování dat poslat cestu souboru a pole měst. Pak se zavolá vlastní funkce a data se připraví pro poslání na výslednou stránku, to je blíže popsáno v kapitole 7.5. Poté se zobrazí výsledná stránka s poslanými daty.

```
def stredocesky(response):
    path = "/svoboda/mysite/static/main/stredocesky.csv"

    Mesta = ["praha/praha-324/", "stredocesky/mlada-boleslav-243/",
    "stredocesky/melnik-232/", "stredocesky/nymburk-284/",
    "stredocesky/kladno-170/", "stredocesky/rakovnik-337/",
    "stredocesky/beroun-11/", "stredocesky/ricany-358/",
    "stredocesky/kolin-176/", "stredocesky/kutna-hora-198/",
    "stredocesky/benesov-9/", "stredocesky/pribram-332/"]

    scrapepocasi(path, Mesta)


    …
    Kód pro poslání dat do výsledné tabulky
    …

    return render(response, "main/stredocesky.html", {"df":df,
    "odkazy":odkazy})
```

Zdrojový kód 23: Ukázka kódu pro středočeský kraj

7.3 Scrapování Komodit

Pro komodity je také potřeba vytvořit funkce pro stránku s formulářem, funkčně je stejná jako stránka pro formulář s počasím, viz Zdrojový kód 24. Pro výsledné tabulky si také vytvořím vlastní funkci, která bude scrapovat data ze stránek různých komodit, jelikož webová stránka vypadá ve všech případech podobně.

```
def Komodity(response):

    if response.method == "POST":
        form = KomodityF(response.POST)
        if form.is_valid():
            vyber = form.cleaned_data["data"]
            return HttpResponseRedirect(vyber)
    else:
        form = KomodityF()

    return render(response, 'main/Komodity.html', {"form":form})
```

Zdrojový kód 24: Kód pro rozcestí komodit

Vlastní funkci jsem pojmenoval **scrapekomodity**. Funkce vyžaduje několik proměnných pro správné fungování. V proměnné `baseurl` se posílá základní URL adresa pro danou komoditu, v proměnné `cestasouboru` se posílá cesta, kde se má vytvořit nebo otevřít daný soubor. V proměnných `jednotky` a `jednotkynazev` se posílají jednotky dané komodity, v jednom případě ve formátu do URL adresy a v tom druhém ve formátu do hlavičky souboru. Poslední proměnnou, kterou je potřeba poslat je proměnná `posun`, ta zaručuje správné stažení datumu

z názvu stránky. Jak je vidět v návrhu na Obrázek 12, měsíc a rok se nacházejí v tagu <h1> až za názvem komodity. Každá komodita má však rozdílnou délku názvu, proto je potřeba tuto délku pro stažení korigovat. Ve Zdrojový kód 25, je vidět základní připravení funkce a funkce pro stažení dat, která je stejná nehledě na to, jestli se stahuje od začátku nebo ne. Celá funkce pro zápis je opatřena odchycením jakýchkoliv chyb. Do proměnných si zapíšu nynější rok a měsíc. Proměnná konec uchovává číslo, kdy se má procházení měsíců zastavit.

Ve Zdrojový kód 26 je pokračování funkce scrapekomodity. Zde se kontroluje, jestli již existuje poslaný soubor a buďto se vytvoří nový soubor, zapíše se hlavička a začne se stahovat nebo se zkontroluje, jaký je poslední zápis a zapíše se pouze zbytek měsíců.

Data v CSV souborech jsou opět oddělována středníkem, důvod v kapitole 7.7.

Problémy

První problém, který jsem objevil, byl v rámci stahovaných dat, viz Zdrojový kód 26. V částkách, které dosahovaly přes tisíce, se nacházela neviditelná mezera a tu potom nebylo možné vložit správně do tabulky. Pomocí knihovny **re** jsem všechna čísla přeformátoval do tvaru bez neviditelné mezery [19].

Druhý problém nebyl v kódu, ale v právech serveru. Jelikož Python otvírá a čte již existující soubor, musel jsem všem staženým souborům dát práva. Pomocí příkazů „chmod 775 /svoboda -R“ a “chown svoboda:www-data /svoboda -R“ jsem nastavil práva a změnil vlastníka všem souborům a složkám ve složce svoboda, kde se nachází můj projekt.

```
def scrapekomodity(baseurl, cestasouboru, jednotky, posun,
    jednotkynazev):
  tz = pytz.timezone("Europe/Prague")
  cas_temp = datetime.now(tz)
  ted_mesic = cas_temp.strftime("%m")
  ted_rok = cas_temp.strftime("%Y")
  konec = 12

  def zapis():
    try:
      with open(cestasouboru, "a", encoding="utf-8") as file:
        zadost = requests.get(url)
        web = zadost.text
        soup = BeautifulSoup(web, "html.parser")
        datum = soup.find("h1").string.split()
        rok = datum[2 + posun]
        mesic = datum[1 + posun]
        tabulka = soup.find("table")
        maximum =
    tabulka.find_all("tr")[1].find_all("td")[1].get_text(strip=True)
        minimum =
    tabulka.find_all("tr")[2].find_all("td")[1].get_text(strip=True)
        prumer =
    tabulka.find_all("tr")[3].find_all("td")[1].get_text(strip=True)
        maximum = maximum.replace(".",",")
        minimum = minimum.replace(".",",")
        prumer = prumer.replace(".",",")

        zacatek =
    tabulka.find_all("tr")[4].find_all("td")[1].get_text(strip=True)
        zaver =
    tabulka.find_all("tr")[5].find_all("td")[1].get_text(strip=True)
        zacatek = re.sub("[^{}]+".format(printable), "", zacatek)
        zaver = re.sub("[^{}]+".format(printable), "", zaver)
        maximum = re.sub("[^{}]+".format(printable), "", maximum)
        minimum = re.sub("[^{}]+".format(printable), "", minimum)
        prumer = re.sub("[^{}]+".format(printable), "", prumer)

        zmena = ((float(zaver)-float(zacatek))/float(zacatek))*100
        zmena = round(zmena, 2)
        zmena = str(zmena).replace(".", ",")


    file.write(f"{rok};{mesic};{prumer};{maximum};{minimum};{zmena}\n")
    except:
            pass
…druhá část ve zdrovém kódu 26…
```

Zdrojový kód 25: Základní nastavení funkce scrapekomodity a funkce zapis

Stejná část Zdrojový kód 25 je použita jak při stahování dat poprvé, tak i při stahování dat pouze na doplnění, viz Zdrojový kód 26.

```
…první část ve zdrojovém kódu 25…
existuje = os.path.exists(cestasouboru)
if existuje == False:
  with open(cestasouboru, "w", encoding="utf-8") as new_file:
    new_file.write(f"Rok;Měsíc;Průměr{jednotkynazev};
Maximum{jednotkynazev};Minimum{jednotkynazev}; Měsíční změna[%]\n")
    for i in range(2008, int(ted_rok)+1):
      if i == int(ted_rok):
        konec = int(ted_mesic) - 1
      for j in range(1, int(konec)+1):
        if int(j) < 10:
          j = "0" + str(j)
        url = baseurl + str(i) + str(j) + str(jednotky)
        zapis()

else:
  with open(cestasouboru, "r", encoding="utf-8") as file:
    radky = file.readlines()
    posledni = radky[len(radky)-1]
    rozdeleny = posledni.split(";")
    posl_rok = int(rozdeleny[0])
    posl_mesic = rozdeleny[1]
    posl_mesic_cislo = int(kalendar[posl_mesic])

  if posl_mesic_cislo == 12:
    start = 1
    posl_rok = posl_rok + 1
  else:
    start = posl_mesic_cislo + 1

  for i in range(posl_rok, int(ted_rok)+1):
    if i == int(ted_rok):
      konec = int(ted_mesic) - 1

    if i == posl_rok + 1:
      start = 1
    for j in range(start, int(konec)+1):
      if int(j) < 10:
        j = "0" + str(j)
        url = baseurl + str(i) + str(j) + str(jednotky)
        zapis()
```

Zdrojový kód 26: Druhá část funkce scrapekomodity

Pro každou komoditu pak vytvořím vlastní funkci, která bude volat tuto obecnou, ve Zdrojový kód 27 je vidět ukázka pro komoditu zlato. Vytvořím si potřebné proměnné a zavolám funkci.

```
def zlato(response):
    base = "https://www.kurzy.cz/komodity/zlato-graf-vyvoje-ceny/"
    cesta = "/svoboda/mysite/static/main/zlato.csv"
    jednotky = "-czk-1g"
    jednotkynazev = "[CZK/g]"
    posun = int(0)
    scrapekomodity(base, cesta, jednotky, posun, jednotkynazev)

    …
    Kód pro poslání dat do výsledné tabulky
    …

    return render(response, "main/zlato.html", {"df":df,
    "odkazy":odkazy})
```

Zdrojový kód 27: Ukázka kódu pro zlato

Stejným způsobem stáhneme data i pro zbylé komodity.

7.4 Scrapování Sportů

Stejně jako u předchozích dvou typů dat, i u sportu je potřeba vytvořit stránku s formulářem a poté stránky pro každou výslednou tabulku. Stránka rozcestí je funkčně podobná jako předchozí, jak je vidět ve Zdrojový kód 28.

```
def Sport(response):

    if response.method == "POST":
        form = SportF(response.POST)
        if form.is_valid():
            vyber = form.cleaned_data["data"]
            return HttpResponseRedirect(vyber)
    else:
        form = SportF()

    return render(response, 'main/Sport.html', {"form":form})
```

Zdrojový kód 28: Kód pro rozcestí sportů

Stránky s tabulkami už nebudou tak jednoduché jako v ostatních případech, v tom smyslu, že bude potřeba pouze jedna funkce pro všechny, ale jelikož jsou tabulky na rozdílných stránkách, tak každé stahování bude jiné. Stránky si lze rozdělit na sporty, jelikož jejich tabulky pochází ze stejných stránek.

V následujících kódech jsou soubory CSV oddělovány středníkem a prováděny změny na desetinné čárky, důvod téhle výměny je popsán v kapitole 7.7.

Basketbal

Pro basketbal jsou pouze dvě tabulky s rozdílnou hlavičkou a velikostí tabulky, proto jsem neměl důvod vytvořit si pro něj vlastní funkci. Hlavičku tabulky jsem se rozhodl napsat ručně jelikož v ní je spousta úprav a je úspornější jí zapsat takhle než přepisovat všechny buňky. V obsahu pak procházím tabulku, vyhazuji nepotřebné sloupce a upravuji formát. Ve Zdrojový kód 29 lze vidět, že to zabralo nejvíce řádků.

```
def BasketbalAktualne (response):
  zadost =  requests.get("https://nbl.basketball/tabulka")
  web = zadost.text
  soup = BeautifulSoup(web, "html.parser")

  with open("/svoboda/mysite/static/main/BasketbalAktualne.csv", "w",
    encoding="utf-8")  as new_file:

    new_file.write("Pořadí;Tým;Zápasy;Výhry;Prohry;Vstřeleno;Obdrženo;Dom
    a;Venku;Výhry Doma;Prohry Doma;Výhry Venku; Prohry Venku")

     for i in range(1, len(soup.find_all("table")[0].find_all("tr"))):

       obsah = soup.find_all("table")[0].find_all("tr")[i].get_text(";",
     strip=True)
       obsah = obsah.replace("Posun v tabulce;", "")
       obsah = obsah.replace(";Popis ikonky\"", "")#vickrat v tabulce
       rozdeleny = obsah.split(";")
       rozdeleny.pop(16)
       rozdeleny.pop(13)
       rozdeleny.pop(8)
       rozdeleny[6] = rozdeleny[6].replace(".", "")
       rozdeleny[7] = rozdeleny[7].replace(".", "")
       rozdeleny.pop(5)
       obsah = ";".join(rozdeleny)

       new_file.write("\n")
       new_file.write(str(obsah))
…
Kód pro poslání dat do výsledné tabulky
…
  return render(response, "main/BasketbalAktualne.html", {"df":df,
"odkazy":odkazy})
```

Zdrojový kód 29: Ukázka kódu pro stahování basketbalové tabulky

Pro historickou tabulku je kód obdobný. URL adresa je stejná, jen se místo první tabulky na stránce, scrapuje druhá tabulka na stránce.

Hokej

V hokeji se už jedná o stejnou tabulku, pouze rozdělenou do tří možností, proto se zde vyplatilo si udělat vlastní funkci na scrapování. Tabulky pro hokej jsou nejlépe připravené pro scraping, jelikož neobsahují žádné loga, grafy ani nic podobného. Jedná se zde pouze o text.

Jak je možné vidět ve Zdrojový kód 30, jediné úpravy jsou pouze ve formátu hlavičky a přepisování desetinného oddělovače. Jelikož hokej má mnoho záznamů v hlavičce, nevejde se popsání všech dat na zobrazovanou stránku, proto nechám nadpisy ve zkratkách a pod tabulku uvedu legendu s vysvětlivkami. Funkce se jmenuje **scrapehokej** a jako argumenty bere číslo tabulky a cestu k souboru, který vytvoří. Čísla pro tabulky jsou 0 pro celou tabulku, 1 pro domácí utkání a 2 pro venkovní utkání, podle pořadí tabulek na stránce.

```
def scrapehokej(cislotabulky, soubor):
  zadost = requests.get("https://www.hokej.cz/tipsport-extraliga/table")
  web = zadost.text
  soup = BeautifulSoup(web, "html.parser")
  base = "/svoboda/mysite/static/main/"
  staticky = base + soubor

  with open(staticky, "w", encoding="utf-8") as new_file:
    tabulka = soup.find_all(name="table", class_ = "table-soupiska")
    hlavicka = tabulka[cislotabulky].find("tr").get_text(";", strip=True)
    hlavicka = hlavicka.replace("B%", "B[%]")
    hlavicka = hlavicka.replace("VPř;", "")
    hlavicka = hlavicka.replace("UOs;", "")
    hlavicka = hlavicka.replace("VV", "VV[%]")
    new_file.write(hlavicka)

    for i in range(1, len(tabulka[cislotabulky].find_all("tr"))):
      obsah = tabulka[cislotabulky].find_all("tr")[i].
            get_text(";", strip=True)
      rozdeleny = obsah.split(";")
      rozdeleny[10] = str(rozdeleny[10].replace(".", ","))
      rozdeleny[11] = str(rozdeleny[11].replace(".", ","))
      rozdeleny[19] = str(rozdeleny[19].replace(".", ","))
      rozdeleny.pop(17)
      rozdeleny.pop(14)
      obsah = ";".join(rozdeleny)

      new_file.write("\n")
      new_file.write(str(obsah))
```

Zdrojový kód 30: Funkce scrapehokej

V tabulkových funkcích pak pouze nastavíme proměnné a zavoláme funkci scrapehokej. Ve Zdrojový kód 31 lze vidět příklad pro první tabulku.

```
def HokejAktualne(response):
    cislo = int(0)
    soubor = "HokejAktualne.csv"
    scrapehokej(cislo, soubor)

    …
    Kód pro poslání dat do výsledné tabulky
    …
    return render(response, "main/HokejAktualne.html", {"df":df,
    "odkazy":odkazy})
```

Zdrojový kód 31: Ukázka pro aktuální tabulku hokejové soutěže

Fotbal

Fotbal, stejně jako hokej, má rozdělené tři tabulky. Oproti hokeji se fotbalové stránky zobrazují na různých adresách, je proto potřeba posílat URL adresu dané stránky. Opět je potřeba si vytvořit vlastní funkci jménem **scrapefotbal** a posílat do ní proměnné url a název souboru. Na začátku si opět nastavím dotaz a zbylé potřebné proměnné. Z důvodu formátování opět hlavičku pouze napíšu a nestahuju, obsah pak stahuju po řádcích.

Ve Zdrojový kód 32 lze vidět oprava prvního problému, kdy při stahování obsahu tabulky se mi data stahovala vždy s prázdným řádkem mezi řádky. Na stránce nic takového vidět nebylo ani po zkontrolování HTML kódu stránky. Z tohoto důvodu jsem se rozhodl kontrolovat, jak je řádek dlouhý. Pokud je řádek prázdný, v mém případě velikost menší než 5, tak se nezapíše. Pokud je plný, odstraní se z pole záznamy na místě grafu a zbytek se zapíše.

```
def scrapefotbal(url, nazev):
  zadost = requests.get(url)
  web = zadost.text
  soup = BeautifulSoup(web, "html.parser")
  base = "/svoboda/mysite/static/main/"
  soubor = base + nazev

  with open(soubor, "w", encoding="utf-8") as new_file:
    tabulka = soup.find(name="table", class_ = "table table--after2022
    table--stage-1")
    new_file.write("Pořadí; Klub; Zápasy; Výhry; Remízy; Prohry; Góly+;
    Góly-; Body")
    for i in range(1, len(tabulka.find_all("tr"))):
      obsah = tabulka.find_all("tr")[i].get_text(";", strip=True)
      obsah = obsah.split(";")
      if len(obsah) < 5:
        pass
      else:
        rozdeleny.pop(11)
        rozdeleny.pop(10)
        rozdeleny.pop(8)
        obsah = ";".join(rozdeleny)
        new_file.write("\n" + obsah)
```

Zdrojový kód 32: Vlastní funkce scrapefotbal

Poté stačí už jen nastavit proměnné a zavolat funkci, viz Zdrojový kód 33. Stejně pak pro domácí a venkovní tabulky.

```
def FotbalAktualne(response):
    url = "https://www.fortunaliga.cz/tabulka/
            2024/aktualni?id_stage=1&order=8&order_direction=2"
    nazev = "FotbalAktualne.csv"
    scrapefotbal(url, nazev)
    …
    Kód pro poslání dat do výsledné tabulky
    …
    return render(response, "main/FotbalAktualne.html", {"df":df,
    "odkazy":odkazy})
```

Zdrojový kód 33: Ukázka pro fotbalovou tabulku

7.5 Zobrazování tabulek a manipulace s daty

Pro zobrazování tabulek jsem se rozhodl pro knihovnu pandas. Knihovna je open-source free software, který zajišťuje funkce pro jednoduché použití datových struktur a datové analýzy. [21]

Prvně je za potřebí si načíst soubor CSV pro dané data. Zdrojový kód 34 ukazuje kód pro soubor s pro komoditu zlato. Po změně typu dat, popsané v kapitole 7.7, uložíme soubor jako XLSX s popisem listu. Poté se načte cesta k souboru XLSX a pokud existuje, tak se nahraje

do proměnné `df`, datový objekt, který slouží pro ukládání buněk tabulek. Dále se vytvoří slovník s odkazy na dané soubory, ten se bude používat pro tlačítka na stažení. Při přepojení na výslednou stránku se pošle jak tabulka, tak i odkazy.

```
…
DoE = pd.read_csv("/svoboda/mysite/static/main/zlato.csv", sep=";")
zmenatypu(DoE, jednotkynazev)
DoE.to_excel("/svoboda/mysite/static/main/zlato.xlsx", "Vývoj ceny
    zlata", index=False)

if os.path.isfile("/svoboda/mysite/static/main/zlato.xlsx"):
  try:
    df = pd.read_excel("/svoboda/mysite/static/main/zlato.xlsx")
  except Exception as e:
    df = None
    print(e)
else:
  df = None
  print("Soubor neexistuje")

odkazy = [
          ("CSV",'main/zlato.csv'),
          ("EXCEL",'main/zlato.xlsx')
         ]

return render(response, "main/zlato.html", {"df":df, "odkazy":odkazy})
```

Zdrojový kód 34: Kód pro poslání dat na výslednou stránku

Následná stránka zpracuje data tak, jak je to popsáno v následující kapitole.

7.6 HTML templaty

Django disponuje systémem pro dědění HTML souborů z jednoho mateřského, v mém případě *base.html*. Soubor *base.html* se nachází ve složce templates, stejně jako ostatní templaty.

První věc, co je potřeba udělat u mateřského souboru, je propojení se statickými soubory. Příkaz se použije úplně jako první, ještě před HTML strukturou. Dále ve Zdrojový kód 35 lze vidět hlavičku souboru base.html.

Systém dědění disponuje možností si stránku rozdělit na bloky, které lze jednoduše přepsat na synovských stránkách. Pokud není žádný blok přepsán, tak se na synovské stránce zobrazí vše tak, jak je to napsané na stránce mateřské. V hlavičce můžeme vidět první využití bloku, a to konkrétně na přepsání titulku stránky.

```
{%load static%}

<!DOCTYPE html>
<html>
<head>
  <title> {%block title %}Hromadná data pro výuku{%endblock%} </title>
  <link rel="icon" type="image/x-icon"
    href="{% static 'main/favicon.ico' %}">
  <link rel="stylesheet" href="{% static 'main/base.css' %}">
</head>
```

Zdrojový kód 35: Hlavička souboru base.html

V těle souboru se pak nachází několik dalších bloků. Jejich chování je vysvětleno v kapitole 3.3. V mém případě používám:

- Blok titulek – zmíněný výše
- Blok zpátky – Nachází se v navigačním baru pouze na finálových stránkách a odkazuje zpátky na rozcestníky
- Blok popis – slouží k popisu aplikace na domovské obrazovce¨
- Blok content – Zde se nachází formuláře stránek a na finálních stránkách pak název tabulky
- Blok stáhnout – na finálních stránkách zobrazuje tlačítka na stažení, na ostatních je prázdný
- Blok tabulka – Nejdůležitější blok, bere data ze souboru views.py a vytvoří z nich tabulku na finální stránce. Celý blok je ohraničen tagem pro lepší aplikování stylů. Pomocí jednoduchých if a for cyklů se dosazují data z poslané proměnné `df`, viz Zdrojový kód 36. Pokud je proměnná prázdná, nic se nestane.

```
<div class="tabulka">
  {% block tabulka %}
  <table>
    <thead>
      <tr>
        {% if not df.empty %}
          {% for sloupec in df.columns %}
            <th>{{ sloupec }}</th>
          {% endfor %}
        {% endif %}
      </tr>
    </thead>
    <tbody>
      {% if not df.empty %}
        {% for radek in df.values %}
          <tr>
            {% for bunka in radek %}
              <td>{{ bunka }}</td>
            {% endfor %}
          </tr>
        {% endfor %}
      {% else %}
        <tr>
          <td colspan="2">Něco se pokazilo. Dejte mi prosím vědět.</td>
        </tr>
      {% endif %}
    </tbody>
  </table>
  {% endblock %}
</div>
```

Zdrojový kód 36: Blok tabulka ze souboru base.html

Posledním blokem na stránce je blok legenda, který slouží pro uvedení zdrojové stránky, ze které čerpám, pod příslušnou tabulku a u hokejové tabulky taky pro popis hlavičky.

Ve všech ostatních synovských HTML souborech musí být na prvním řádku kód pro dědění ze souboru *base.html*. Ve Zdrojový kód 37, je ukázka souboru pro počasí v Jihočeském kraji. Přepisují se pouze ty bloky, které potřebujeme.

```
{% extends "main/base.html" %}
{% block title %}Jihočeský kraj{%endblock%}

{% block zpatky%}<a href="/hromadna_data/Pocasi">Zpět na
    rozcestí</a>{%endblock%}

{%block content%}
<span class="final">Tabulka počasí pro Jihočeský kraj</span>
{%endblock%}

{% block legenda %}
 Data jsou čerpaná ze stránky: https://www.in-pocasi.cz/
{% endblock %}
```

Zdrojový kód 37: soubor jihocesky.html

7.7 Problémy při implementaci

Dílčí problémy, které vznikly během psaní kódu pro scrapování jsou uvedeny ve svých kapitolách. Dále se zaměříme na více obecné problémy, které se během psaní objevily a které bylo nutné odstranit.

Český excel nepracuje s desetinou tečkou

Desetinná tečka se používá v anglickém formátu čísla a český excel s ní neumí pracovat. Jelikož je aplikace pro české školy, které budou mít excel nastavený do češtiny a českého formátu, je nutné tomu aplikaci přizpůsobit. Je potřeba přepsat všechna čísla, která mají desetinou tečku na čárku. Jedná se o teplotu v počasí, procentní úspěšnosti v hokejových tabulkách a všechny ceny u komodit. Z tohoto důvodu je však nutné změnit taky zapisování dat tak, aby se data oddělovala středníkem a ne čárkou. Aby si uživatel nemusel dohledávat, která data jsou oddělena čárkou a která středníkem, rozhodl jsem se všechny soubory CSV oddělovat středníkem, i když tam problém s tečkou a čárkou nebyl.

Ve Zdrojový kód 38 jsou ukázky upraveného kódu pro počasí. Je zde vidět prvotní zápis hlavičky, přepisování stažené teploty a poté formátování proměnné `data`, která se zapisuje do souboru. Takové změny bylo zapotřebí udělat u všech výše zmiňovaných typů dat. U všech políček, která mají v sobě nějaký znak, například procento nebo zde stupně, bylo nutné stupně odstranit, aby buňka byla pouze číslo a bylo možné ji jednoduše přepsat na číslo.

```
…
nove.write("Den;Čas;Město;Nadmořská výška[m.n.m.];Teplota[°C];Pocitová
    teplota[°C];Vlhkost[%]\n")
…
teplota = soup.find(name="div", class_="alfa mb-1").string.strip()
teplota = teplota.replace(".", ",")
teplota = teplota.replace("°C", " ")
…
data =
    f"{den};{cas};{mesto};{nadMvyska};{teplota};{pocit};{vlhkost}"+"\n"
```

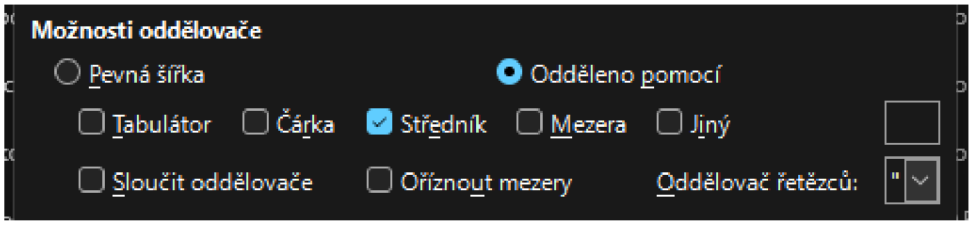
Zdrojový kód 38: Upravené zápisy z funkce scrapepocasi

Tento postup však nebyl snadno proveditelný. Při otevření XLSX souboru v aplikaci Excel, se sice číslo zobrazilo, ale bylo uložené jako text a nebylo možné s ním dále pracovat. Pokusil jsem se v kódu upravit buňky jednoduše pomocí základní funkce Pythonu na změnu typu dat, například *teplota = float(teplota)*. Tohle ovšem nefungovalo, opět z důvodu rozdílného jazykového nastavení, Python nedokázal změnit string na float s desetinnou čárkou. Použil jsem proto knihovnu **locale**, která umožňuje práci s lokalizací [20]. Prvně je potřeba knihovnu importovat do souboru *views.py*. Poté jsem si nastavil lokalizaci pro čísla. Nakonec, ve Zdrojový kód 39, je vidět vytvoření vlastní funkce na změnu dat pro komodity. Tahle změna zasahuje do kódu z kapitoly 7.5., kde po načtení souboru pomocí knihovny pandas, místo konvertování souboru ihned na typ XLSX, se prvně změní typ dat. V kódu lze vidět proměnnou `pandi`, ve které se nachází načtený CSV soubor. Druhá proměnná využívá název jednotek použitý pro danou komoditu, jelikož se musí uvést přesný název sloupce. U vybraných sloupců se poté zavolá funkce z knihovny **locale** pro změnu typu dat v daném sloupci.

```
import locale
…
locale.setlocale(locale.LC_NUMERIC, 'cs_CZ.UTF-8')
…
def zmenatypu(pandi, jedn):
    pandi[f'Průměr{jedn}'] = pandi[f'Průměr{jedn}'].apply(locale.atof)
    pandi[f'Maximum{jedn}'] = pandi[f'Maximum{jedn}'].apply(locale.atof)
    pandi[f'Minimum{jedn}'] = pandi[f'Minimum{jedn}'].apply(locale.atof)
```

Zdrojový kód 39: Použití knihovny locale

Pro všechna ostatní data je řešení podobné, jen s jinými názvy. Po téhle úpravě již všechna čísla v souborech XLSX, otevíraných v Excelu, fungují správně. Pro soubory CSV je ale potřeba při otevírání nastavit oddělovač středník. Například v LibreOffice je jako základní oddělovač čárka, ale při každém otevření souboru se zobrazí možnost volby, viz Obrázek 17.



Obrázek 17: Nastavení oddělovače v LibreOffice

8 Nasazení

První verze aplikace jsem měl zprovozněné na dočasném web hostingovém serveru od firmy Linode, kde jsem získal zkušební verzi na tři měsíce zdarma. Při přesunu na školní server bylo nutné udělat několik oprav.

Za prvé, do složky s projektem bylo potřeba přidat textový soubor, který obsahuje všechny potřebné knihovny a moduly, které musí Python stáhnout. Pomocí příkazu **py -m pip freeze > requirements.txt** se vytvoří soubor a do něj se zapíšou všechny používané balíčky s jejich verzemi. Poté jen stačí nainstalovat všechny balíčky příkazem **python3 pip install -r requirements.txt.** Mít k dispozici takovýto soubor se hodí v případech přesouvání projektu na jiné servery. Ve Zdrojový kód 40 je ukázáno pár řádků mého souboru knihoven.

```
beautifulsoup4==4.12.2
Django==4.2.2
pandas==2.1.1
pytz==2023.3.post1
requests==2.31.0
…
```

Zdrojový kód 40: Ukázka ze souboru requirements.txt

Druhý problém nastal se složkou static, kam se ukládají veškeré soubory. Cesta na tuto složku, která fungovala na testovacím serveru nefungovala na tom školním. Na testovacím serveru stačilo odkazovat na soubory pomocí **static/soubor**. Na školním serveru je potřeba odkazovat na soubory plnou cestou **/svoboda/mysite/static/main/soubor**. Po zprovoznění bylo potřeba dát všem složkám práva na čtení souborů, stejné jako popisuju v kapitole 7.3.

Během vývoje se Django stará o veškeré statické soubory, které je potřeba zobrazit a pracuje s nimi. Po ukončení vývoje je však potřeba nastavit, aby se o tyhle soubory staral někdo jiný, v mém případě Apache. Na serveru v souborech Apache, je potřeba nakonfigurovat soubor *apache2.conf*, pro specifikaci, se kterými soubory má pracovat. Tento soubor nalezneme na cestě **/etc/apache2/apache2.conf**. Do souboru se přidají následující řádky, viz Zdrojový kód 41. První tři řádky propojují Apache a mojí aplikaci, konkrétně soubor WSGI, virtuální prostředí Djanga a základní cestu mé aplikace. Kód pod tagem Directory umožňuje přístup k souboru v plném rozsahu. Alias static poté ukazuje na složku, kde se nachází statické soubory aplikace.

```
WSGIScriptAlias / /svoboda/mysite/mysite/wsgi.py
WSGIPythonHome /svoboda/mysite/myprojectenv
WSGIPythonPath /svoboda/mysite
<Directory /svoboda/mysite/mysite>
    <Files wsgi.py>
    Require all granted
    </Files>
</Directory>
Alias /static/ /svoboda/mysite/static/
<Directory /svoboda/mysite/static>
    Require all granted
</Directory>
```

Zdrojový kód 41: Soubor apache2.conf

Cestu ke statickým souborům je potřeba nastavit i v souboru *settings.py* v Djangu, viz Zdrojový kód 42. V souboru *settings.py* je také potřeba vypnout DEBUG mód. Pokud je tento mód vývoje zapnutý, stránka v případě chyby poskytne větší množství informací, které mohou pomoci v opravě chyby. Pokud je tento mód vypnutý, stránka zobrazí klasickou chybu. Tímto se předkládání statických souborů stane úkolem nastaveného Apache.

```
DEBUG = False
…
STATIC_URL = '/static/'
STATIC_ROOT = "svoboda/mysite/static/"
```

Zdrojový kód 42: Soubor settings.py

9 Testování a ladění

Všechna má testování probíhala na hodinách informatiky na Základní škole Tylova v Písku v rámci asistentských pedagogických praxí. Třídy jsou na hodinách informatiky rozdělené do dvou polovin. Škola má zakoupenou licenci Office 365, proto testování probíhalo v aplikaci Excel, konkrétně Excel 2016.

9.1 Alfa Testování

První alfa testování proběhlo ve třídě 8.B, kde jsem namísto plánovaných dvaceti minut vyučoval celou hodinu.

9.1.1 Úlohy pro testování

Pro otestování mé aplikace jsem si vytvořil krátkou sadu úloh čerpající ze všech typů dat. Úkoly byly následující:

1. Stáhněte si data počasí pro vámi libovolně zvolený kraj. Soubor si uložte a nechte být, vrátíte se k němu později.
2. Vytvořte graf ceny elektřiny za období 2012–2020. Zjistěte nejdražší měsíc.
3. Ve vývoji cen zlata vypočítej měsíční změnu v procentech. O kolik se zvětšila nebo zmenšila průměrná cena oproti předchozímu měsíci?
4. Seřaďte tabulku hokejových týmů podle daných gólů v přesilovkách. Seřaďte tabulku hokejových týmů podle obdržených gólů v přesilovkách.

   Nacházejí se na prvních pozicích stejné týmy? Odůvodni.
5. Znovu si stáhněte počasí pro stejný kraj jako v úkolu č. 1. Tabulky si zkopírujte do jednoho souboru. Zjistěte průměrnou změnu teploty ve městech vašeho kraje.

   Pokud nenastala změna, dostanete od učitele jiný soubor.

V tuto chvíli jsem si nebyl jist po jak dlouhé době se teplota na stránkách aktualizuje, jelikož stránky uvádí aktualizaci vždy v celou hodinu. Měl jsem proto připravený soubor se všemi kraji stažený o pár hodin dříve.

9.1.2 Poznatky po testování

Při stahování počasí se většině žákům soubory otevíraly v online verzi Excelu a na stránce se pouze objevil proužek s tlačítkem pro další stažení. Je tedy nutné upozornit žáky, že mají soubory stahovat, a ne otevírat v online verzích aplikace. Toto je problém především prohlížeče Microsoft Edge, při používání prohlížeče Google Chrome se soubory stahovaly rovnou.

Dále je potřeba žáky upozornit na pomalejší stahování dat. Žáci byli nedočkaví, a to i v případě stahování počasí jednotlivých krajů, které zabralo v zatížení celou třídou maximálně deset vteřin. Pokud žáci kliknou na znovunačtení stránky, může dojít k poškození stahování a některá města se ve výsledné tabulce objeví vícekrát. Když jsem se tento problém snažil napodobit, tak se mi to nepodařilo, nejspíš byl problém ve více uživatelích, snažících se dostat stejná data.

Jak potvrdila praxe, je nutné žákům zdůraznit že všechna data naleznou na uvedené stránce a není proto žádoucí, aby je vyhledávali ručně na Googlu.

Větší problém nastal při stahování cen elektřiny. Před testováním jsem ještě prováděl několik změn a v rámci toho jsem odstranil již vytvořené soubory pro komodity. První stažení komodit vždycky trvá déle, jelikož se musí projet každý měsíc historie zvlášť. Problém nastal, když několik žáků současně zadalo příkaz k vyhledávání cen elektřiny. V přístupu, který jsem zvolil, první stažení vytvoří soubor a zapisuje všechny měsíce. Každé další stažení pak otevře již existující soubor a připíše zbylé měsíce. V tomhle případě tedy první žák začal stahovat historii celou, další žádost o stažení si soubor otevřela po úspěšném stažení přibližně jednoho roku a začala stahovat zbylé měsíce od této doby. Ve výsledku se na server dostalo příliš mnoho žádostí současně, což server zahltilo. Mezitím jsem dětem poslal soubor s počasím, aby splnili alespoň úkol číslo 5. Poté, co server ukončil stahování cen elektřiny, se ve výsledné tabulce nacházelo několik hodnot vícekrát. Například každý měsíc z roku 2023 byl v tabulce šestkrát. Bez zásahu do kódu tomu lze zamezit tím, že si učitel předem projde všechny stránky komodit a tím aktualizuje tabulku sám. Žákům se pak bude ukazovat celá tabulka bez žádného scrapingu navíc. Na to ale nemůžeme spoléhat, proto jsem kód upravil, viz kapitola 9.2.

Po obnovení činnosti serveru jsem žákům ukázal špatný výstup ohledně cen elektřiny a řekl jim, aby použili data o cenách zlata i pro splnění úkolu číslo 2 a pak až pracovali na dalších úlohách. Během kontroly úkolu ze stahování dat o počasí, jsem si všiml, že i tabulky, které byly stažené sousedními žáky v rozmezí dvou minut, mají jiné hodnoty teplot. Nebyly to časy přes hodinu, ale v rámci jedné hodiny, konkrétně 11:03 a 11:05. Stránky s počasím se tedy aktualizují i vícekrát za hodinu, nejen v celou, jak na stránkách píší.

Vzhledem ke zpoždění, které nastalo z výše zmíněného důvodu žáci nestihli splnit všechny úkoly a skončili s rozpracovaným úkolem 2.

Jedna dvojice žáků přeskočila úkol číslo 3 a začala pracovat na úkolu číslo 4. Zde jsem se setkal s problémem špatně zadaného úkolu. V zadání je uvedeno, že chci tabulky seřadit podle vstřelených a obdržených gólů v přesilovkách. Bohužel jsem toto špatně formuloval, v tabulce se nenachází data o obdržených gólech týmu, který má přesilovku, ale jsou to data o obdržených

gólech týmu, který má oslabení. Po objasnění už nebyl problém. Pro příští testování budu úkol lépe formulovat.

9.2 Ladění po alfa testování

Jelikož se nelze spoléhat na uživatele, aby otevřel soubor za správných podmínek, je potřeba odstranit veškeré chyby z alfa testování.

Prvně jsem se rozhodl řešit kontrolu doby, kdy byl soubor naposledy modifikován a potom až jestli se můžou data zapisovat. Při řešení ale vyšlo najevo, že je potřeba to provést současně.

Do souboru *views.py* jsem proto přidal nové funkce pro kontrolu času modifikací a pro kontrolu zápisu. Ve funkci kontrola času srovnávám čas poslední modifikace s nynějším časem, výsledek se pak porovná s argumentem délka, který se posílá při zavolání v minutách. Pokud je rozdíl větší než daný časový úsek, pak funkce vrátí True, potřebu stažení souboru. Pokud je časový úsek menší, funkce vrátí False, není potřeba stahovat znovu. Pokud soubor ještě nebude existovat, funkce vrátí True, viz Zdrojový kód 43. Tímto se zrychlí zobrazování souborů, všem ostatním žákům kromě prvního.

```
def kontrola_casu(cesta, delka):
    try:
        soubor = os.path.getmtime(cesta)
        citelny = datetime.fromtimestamp(soubor)
        cas_ted = datetime.now()
        zmena = cas_ted - citelny
        usek = timedelta(minutes=delka)
        if zmena > usek:
            return True
        else:
            return False
    except FileNotFoundError:
        return True
```

Zdrojový kód 43: Funkce pro kontrolu času

Další funkce se týkají zamykání souboru, aby do něj mohl zapisovat pouze jeden uživatel. Vytvořil jsem funkci **kontrola_zamek**, která bere jako argument cestu k souboru a z něj vytvoří cestu k souboru, který slouží jako zámek. Pokud takový soubor neexistuje, pouze při prvotních staženích, vytvoří ho. Textové soubory s názvem **upravuju (typ dat).txt** v sobě obsahují jediný znak, a to jedničku nebo nulu, znázorňující povolení zapisovat. Funkce vrací hodnoty True nebo False podle stavu zámku, viz Zdrojový kód 44.

```
def kontrola_zamek(cestasouboru):
    zamek = cestasouboru.split("/")
    typ = zamek[len(zamek)-1].replace(".csv", "")
    zamek[len(zamek)-1] = f"upravuju_{typ}.txt"
    zamek = "/".join(zamek)
    try:
        with open(zamek, "r") as file:
            radek = file.readlines()
            povoleni = int(radek[0])

            if povoleni == 1:
                return True
            else:
                return False
    except FileNotFoundError:
        with open(zamek, "w") as file:
            file.write("1")
        return kontrola_zamek(cestasouboru)
```

Zdrojový kód 44: Funkce kontrola_zamek

K této funkci jsem si ještě vytvořil dvě další funkce pro zamknutí a odemknutí zámku. Funkce pouze mění číslo v souboru na požadovanou hodnotu.

Ve funkcích pro zobrazovaní stránky je pak potřeba přidat volání těchto funkcí. V následujícím Zdrojový kód 45 je vidět implementace pro data počasí v Jihočeském kraji. Do proměnných se zavolají výše zmíněné funkce. Pokud soubor nebyl změněn delší dobu než daná hodnota, u počasí 5 minut, zkontroluje se, jestli někdo nezapisuje. Pokud soubor nebyl delší dobu změněn, mělo by to znamenat, že do něj nikdo aktuálně nezapisuje, pro jistotu je tam ještě druhá kontrola zámku. Když se dostane povolení k zápisu, daný proces zamkne soubor, stáhne data a poté soubor zase odemkne. Pokud v jakékoliv fázi je procesu zamítnut zápis do souboru, uživatel je poslán na čekací stránku.

```
kontrolaCas = kontrola_casu(path, 5)
zapis = kontrola_zamek(path)
if kontrolaCas == True:
  if zapis == True:
    zamknout(path)
    scrapepocasi(path, Mesta)
    odemknout(path)
    … kód pro změnu typu dat…

  else:
    return HttpResponseRedirect("priprava/jihocesky")
elif zapis == False:
  return HttpResponseRedirect("priprava/jihocesky")
else:
  pass
```

Zdrojový kód 45: Ukázka kontrol pro jihočeský kraj

Pro tabulky vybraných sportů a cen komodit je implementace obdobná. Pro sport jsem čas obnovy zvolil 180 minut. Komodity potřebují aktualizovat ještě méně frekventovaně. Zvolil jsem 7 hodin, aby si učitel případně mohl data připravit před první hodinou v první den nového měsíce.

Na čekací stránce je uživatel upozorněn na to, že data se stahují a že musí chvíli počkat. Musel jsem si proto vytvořit nový template *priprava.html* a přidat propojení do souboru *urls.py*. Čekací stránka využívá dynamickou URL adresu, abych mohl snadno zjistit, odkud uživatel přišel a nabídnout mu tlačítko pro další pokusy. V soboru *urls.py* je tedy spojení navázáno tímto řádkem: **path("priprava/<odkud>", views.priprava, name="priprava")**. Jakýkoliv text za lomítkem přípravy je poslán jako proměnná dál.

V souboru *views.py* je potřeba si udělat funkci pro zobrazení stránky. Funkce pouze vezme data z proměnné `odkud` a pošle je dál do HTML templatu jako proměnnou `z`, viz Zdrojový kód 46.

```
def priprava(response, odkud):
    return render(response, "main/priprava.html", {"z":odkud})
```

Zdrojový kód 46: Funkce priprava

V HTML souboru poté používám poslanou proměnnou v tlačítku, aby se mohl uživatel jedním kliknutím dostat na požadovanou tabulku, viz Zdrojový kód 47. Pokud se tabulka stále nenačetla, uživatel zůstává na čekací stránce.

```
{% extends "main/base.html" %}
{% block title %}Soubory se připravují{%endblock%}

{%block content%}
<div id="priprava">
<p> Jiný uživatel právě stahuje data. Soubory se pro Vás připravují,
    prosím vyčkejte a zkuste to za minutku znovu. <br>
    </p>

    <a href="https://django.pf.jcu.cz/hromadna_data/{{z}}">
        <button type="button">Zkusit znovu!</button>
    </a>
</div>
{%endblock%}

{% block stahnout %}
{% endblock %}
```

Zdrojový kód 47: Soubor priprava.html

9.3 Beta testování

Další testování proběhlo o několik dní později, ve třídě 8.A. Úkoly byly stejné jako při alfa testování, pouze s upraveným zadáním, aby bylo lépe pochopitelné. Poté jsem prohodil pořadí úkolů, aby se v případě pomalé práce stihla alespoň druhá polovina, která nebyla vyzkoušena při alfa testování.

Z testování již nevzešly žádné problémy a aplikace fungovala tak, jak měla. Rychlejší žáci zvládli dokončit všechny úkoly.

10 Závěr

Výsledkem mé práce je vytvoření webové aplikace, která shromažďuje data, připravená k výuce hromadného zpracování dat. Přínos mé práce spočívá ve vytvoření výukové pomůcky, kterou mohou vyučující použít pro zrychlení přípravy na výuku. Data na stránce jsou připravena k použití v tabulkových procesorech. V rámci nalezení limitů a možností jazyka Python, se nevyskytly žádné konkrétní problémy, které by nebylo možné vyřešit. Aplikace běží na školním serveru na adrese https://django.pf.jcu.cz/.

Aplikace je nyní limitována typy dat, které ale lze snadno rozšířit. Data jsem vybíral podle toho, na co by mohla být použita. Vývoj cen komodit jsem vybral pro práci s grafy, informace o počasí především pro tabulkové funkce a sportovní tabulky pro podmíněné formátování. S odstupem času bych do aplikace přidal více dat, abych zachoval rozmanitost. Například je možné přidat k počasí i pravděpodobnost srážek, což by mohlo sloužit také k tabulkovým funkcím či podmíněnému formátování.

Na moji práci by mohla navazovat činnost, která by vytvářela sadu úloh pro výuku přímo z mé aplikace. Další práce by mohly dále zkoumat limity Pythonu za pomoci jiného frameworku, popřípadě za pomoci Djanga v jiném kontextu.

Seznam použité literatury a zdrojů

[1] *Version 2024.2.1. Pyscript documentation* [online]. 2024 [cit. 2024-02-24]. Dostupné z: https://pyscript.github.io/docs/2024.2.1/

[2] *Version 3.0.0 Flask documentation* [online]. 2023 [cit. 2024-02-24]. Dostupné z: https://flask.palletsprojects.com/en/3.0.x/

[3] DJANGO SOFTWARE FOUNDATION. *Version 5.0 Django documentation* [online]. 2024 [cit. 2024-02-24]. Dostupné z: https://docs.djangoproject.com/en/5.0/

[4] KORSUN, Julia. 10 Popular Django Websites That You Probably Know, *DjangoStars* [online]. Wilmington: DS Technologies, 2023 [cit. 2024-03-04]. Dostupné z: https://djangostars.com/blog/10-popular-sites-made-on-django/

[5] DJANGO SOFTWARE FOUNDATION. Writing your first Django app, part 1, *Django documentation,* [online]. 2024 [cit. 2024-03-20]. Dostupné z: https://docs.djangoproject.com/en/5.0/intro/tutorial01/

[6] DJANGO SOFTWARE FOUNDATION. URL dispatcher, *Django documentation* [online]. [cit. 2024-03-14]. Dostupné z: https://docs.djangoproject.com/en/5.0/topics/http/urls/

[7] DJANGO SOFTWARE FOUNDATION. Writing views, *Django documentation* [online]. [cit. 2024-03-14]. Dostupné z: https://docs.djangoproject.com/en/5.0/topics/http/views/

[8] DJANGO SOFTWARE FOUNDATION. Django shortcut functions, *Django documentation* [online]. [cit. 2024-03-14]. Dostupné z: https://docs.djangoproject.com/en/5.0/topics/http/shortcuts/

[9] DJANGO SOFTWARE FOUNDATION. The Django template language, *Django documentation* [online]. [cit. 2024-03-14]. Dostupné z: https://docs.djangoproject.com/en/5.0/ref/templates/language/

[10] DJANGO SOFTWARE FOUNDATION. Writing your first Django app, part 6, *Django documentation* [online]. 2024 [cit. 2024-03-28]. Dostupné z: https://docs.djangoproject.com/en/5.0/intro/tutorial06/

[11] DJANGO SOFTWARE FOUNDATION. Working with forms, *Django documentation* [online]. 2024 [cit. 2024-03-18]. Dostupné z: https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it/https://docs.djangoproject.com/en/5.0/topics/forms/#get-and-post

[12] KIRSTEN, S. *Cross Site Request Forgery (CSRF)* [online]. 2024 [cit. 2024-03-20]. Dostupné z: https://owasp.org/www-community/attacks/csrf#overview

[13] ROYCE, Winston W. Managing the development of large software systems. *Proceedings of IEEE WESCON* [online]. 1970 [cit. 2024-02-17]. Dostupné z: https://web.archive.org/web/20190307012048/http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf

[14] VANÍČEK, J. a BERKI J. *Modelový ŠVP: OPATRNĚ VPŘED* [online]. 2023 [cit. 2024-04-09]. Dostupné z: https://www.imysleni.cz/images/SVP/SVP1_opatrne-vpred_v20230323.pdf

[15] NÁRODNÍ SPORTOVNÍ AGENTURA. Seznam sportovních organizací s počty sportovců a trenérů, *REJSTŘÍK SPORTU Národní sportovní agentura* [online]. 2024 [cit. 2024-03-07]. Dostupné z: https://rejstriksportu.cz/dashboard/public/agenda/sportovciTreneri

[16] NÁRODNÍ SPORTOVNÍ AGENTURA. *Výsledky výzkumů: Míry popularity sportu v České republice 2020, Finanční náročnosti provozování sportů z hlediska pořízení potřebné výstroje a výzbroje* [online]. 2020 [cit. 2024-03-07]. Dostupné z: https://nsa.gov.cz/wp-content/uploads/2021/01/Vyzkumy_popularita_a_fin_narocnost_sportu_2020-2.pdf

[17] hakiran78. What is Web Scraping and How to Use It? *GeeksforGeeks* [online]. 2024 [cit. 2024-03-15]. Dostupné z: https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it/

[18] BISHOP, Stuart. *Pytz - World Timezone Definitions for Python* [online]. 2008 [cit. 2024-03-03]. Dostupné z: https://pythonhosted.org/pytz/

[19] PYTHON SOFTWARE FOUNDATION. Re — Regular expression operations, *python* [online]. [cit. 2024-03-04]. Dostupné z: https://docs.python.org/3/library/re.html

[20] PYTHON SOFTWARE FOUNDATION. Locale — Internationalization services, *python* [online]. [cit. 2024-03-08]. Dostupné z: https://docs.python.org/3/library/locale.html

[21] PANDAS. Pandas documentation, *pandas* [online]. 2024 [cit. 2024-03-08]. Dostupné z: https://pandas.pydata.org/docs/