Univerzita Hradec Králové

Fakulta Informatiky a management

Katedra informatiky a kvantitativních metod

Webové frameworky pro Python

Bakalářská práce

Autor: David Malík

Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. Mgr. Tomáš Kozel, Ph.D.

Hradec Králové duben 2024

Prohlášení

Prohlašuji, že jsem bakalářskou práci na téma „Webové frameworky pro Python“ zpracoval samostatně a s použitím uvedených zdrojů.

.....

V Hradci Králové dne 19. 4. 2024 David Malík

Poděkování

Rád bych poděkoval svému vedoucímu bakalářské práce, doc. Mgr. Tomášovi Kozlovi, Ph.D., za jeho vedení, odborné připomínky a cenné rady, které mi pomohly při zpracování této práce.

Abstrakt

Cílem práce je seznámení s frameworky pro tvorbu webových aplikací v Pythonu a jejich porovnání. Bakalářská práce začíná úvodem do programovacího jazyka Python. Následně je představen rozdíl mezi frameworky a knihovnami. Další kapitola se zabývá úvodem do webových aplikací. Popisuje frontendové technologie pro tvorbu uživatelského rozhraní aplikací a zároveň uvádí do problematiky backendových technologií a databázových systémů. Poté se práce věnuje představení jednotlivých frameworků. Ty jsou v další části porovnány na základě kvality dokumentace, metod zabezpečení, frekvence aktualizací, sociálních metrik a také výkonu, který je testován pomocí load testů a spike testů. Závěrečná kapitola práce se zabývá zhodnocením výsledků porovnání jednotlivých frameworků. Zahrnuje také doporučení pro výběr vhodného frameworku na základě specifických potřeb projektu a preferencí vývojáře.

Klíčová slova: webová aplikace, Python, framework, porovnání, Django, Flask, Tornado, CherryPy, Pyramid, Bottle

Abstract

Title: Web frameworks for Python

The aim of the thesis is to introduce frameworks for creating web applications in Python and to compare them. The bachelor's thesis begins with an introduction to the Python programming language. Next, the difference between frameworks and libraries is presented. The next chapter deals with an introduction to web applications. It describes frontend technologies for creating user interfaces of applications and also introduces the issues of backend technologies and database systems. Then, the thesis is devoted to the presentation of individual frameworks. These are compared in the next part based on the quality of documentation, security methods, update frequency, social metrics, and also performance, which is tested using load tests and spike tests. The final chapter of the thesis deals with the evaluation of the results of comparing individual frameworks. It also includes recommendations for choosing a suitable framework based on the specific needs of the project and the developer's preferences.

Keywords: web application, Python, framework, comparison, Django, Flask, Tornado, CherryPy, Pyramid, Bottle

Obsah

Seznam obrázků

Seznam kódů

Seznam grafů

Seznam tabulek

1. Úvod

Trendem dnešní doby je být neustále připojen do internetového prostředí. Lidé si na internetu běžně pouštějí písničky přes streamovací platformy, sledují videa různého druhu, hledají inzeráty se zajímavými produkty, seznamují se na seznamkách a sdílejí své každodenní zážitky. Všechny tyto věci jsou spojené s termínem webové aplikace. Tyto aplikace využívá denně velká část lidské populace, ale pouze malá část zná principy, jakým způsobem fungují a nástroje, kterými je lze tvořit. Existuje velké množství nástrojů pro tvorbu webových aplikací, které se liší svými funkcemi, výkonností a kvalitou dokumentace. Některé z nich jsou určeny pro rychlý vývoj prototypů, zatímco jiné poskytují rozsáhlé knihovny pro vytváření komplexních systémů. S rostoucím počtem zařízení připojených k internetu a neustálým vývojem technologií je možné očekávat, že se množství dostupných nástrojů bude nadále rozšiřovat a vyvíjet.

2. Cíl práce

Cílem této bakalářské práce je představit jednotlivé frameworky pro tvorbu webových aplikací pomocí programovacího jazyka Python. Práce se zaměří na srovnání klíčových vlastností a výkonu těchto frameworků tak, aby poskytla ucelený přehled o jejich možnostech a omezeních.

3. Python

Python je objektově orientovaný[1] interpretovaný[2] dynamický[3] programovací jazyk, který využívá virtuální stroj PVM. [1] Jednou z vlastností programovacího jazyka Python je jeho modularita, která umožňuje organizovat kód do souborů nazývaných moduly. Ty obsahují definice funkcí a příkazy. Python nabízí řadu standardních modulů. [2] Jeden takový se nazývá sys a obsahuje informace o operačním systému, jako například, která verze Pythonu je používána. [3] Python je multiplatformní, dokáže tedy běžet na různých operačních systémech jako je Linux, Windows a MacOS, avšak instalace a nastavení se na různých platformách liší. [4] Python je široce využíván pro vývoj webových i desktopových aplikací, datovou vědu a strojové učení. Mezi vývojáři je oblíbený především pro snadné naučení a efektivní psaní kódu. [5]

3.1. PVM (Python Virtual Machine)

Programovací jazyk Python je jazyk s virtuálním strojem. To znamená, že svůj kód nepřevádí přímo do strojového kódu. Nejprve je zdrojový kód Pythonu kompilérem[4] převeden do mezikódu nazývaného ByteCode. ByteCode má příponu .pyc a je uložen v úložišti počítače. Poté je tento kód předán virtuálnímu stroji PVM, který funguje jako interpretr. Kód je následně pomocí PVM převeden z ByteCode do strojového kódu. Nedošlo-li během konvertování k chybě, tak je dále strojový kód předán procesoru. V opačném případě je převod zastaven a je zobrazena chybová hláška. [6]

---

[1] **Objektově orientovaný –** programovací paradigma s důrazem na objekty a třídy.
[2] **Interpretovaný –** kód je prováděn řádek po řádku.
[3] **Dynamický –** programovací jazyk, který určuje typ proměnných za běhu podle jejich hodnot.
[4] **Kompilér –** program, který převádí celý zdrojový kód do spustitelného kódu.

3.2. Historie Pythonu

První verze programovacího jazyka Python byla vydána 20. února roku 1991 holandským programátorem, který se jmenoval Guido van Rossum. Ten na Pythonu začal pracovat během vánočních svátků a dále ho vyvíjel ve svém volném čase. Název Python mu dal na základě komediální série od BBC s názvem Monty Python's Flying Circus. [7]

3.3. Python 2.x vs Python 3.x

Python během své existence dostal mnoho aktualizací, přičemž nejvýznamnější byl přechod z verze 2 na verzi 3, který přinesl řadu podstatných změn. Verze 2.0 programovacího jazyka Python byla vydána v roce 2000 a její poslední podverze 2.7 byla vydána v roce 2010. Verze Pythonu 3, která následovala po svém předchůdci, byla uvedena na trh v prosinci 2008. Ta přišla s mnohými úpravami, mezi ně patří změna syntaxe, tak aby byla lépe čitelná a pochopitelná. Došlo ke zlepšení výkonu, což umožnilo dosáhnout stejných výsledků v kratším čase. Změnou prošlo i celočíselné dělení, které v případě verze 3 již nevrací zkrácené desetinné číslo v datovém typu int, ale vrací vždy výsledek typu float. V roce 2020 došlo k ukončení podpory poslední vydané verze Pythonu 2. Dnes existuje už pouze malé množství projektů vyvíjených ve verzi 2, jedná se hlavně o starší udržované projekty. [8]

3.4. Využití Pythonu

V dnešní době nachází programovací jazyk Python uplatnění v několika významných technologických odvětvích. Mezi nejvýznamnější patří analýza a vizualizace dat. Na to pak navazuje umělá inteligence, kde jsou zpracovaná data dále využita k trénování jejich modelů pro specifický úkol. Dalším velice výrazným odvětvím, kde Python nachází své uplatnění, je vývoj webových aplikací. K jejich vývoji se v programovacím jazyce Python v dnešní době nejvíce využívají např. frameworky Django a Flask. Kromě webových aplikací a práci s daty, lze Python velice užitečně využít k automatizaci opakujících se úloh. Python také nachází uplatnění v tvorbě botů[5]. Vzhledem k jednoduché syntaxi a dostupným knihovnám pro web scraping, což je extrakce dat ze stránek, je Python skvělou a rychlou volbou pro získávání dat. [9]

[5] **Bot –** software vykonávající nějakou automatizovanou činnost

4. Framework

V kontextu softwarového inženýrství se za framework považuje kolekce znovupoužitelných softwarových komponent, které umožňují efektivnější vývoj nových aplikací. Frameworky obsahují opakovaně použitelné moduly kódu (knihovny), které jsou založené na konkrétních softwarových standardech a protokolech. Využívání frameworků může pomoci s urychlením vývoje, snížením duplicity kódu a zlepšením jeho čitelnosti. [10]

4.1. Framework vs knihovna

Frameworky i knihovny fungují na principu znovupoužitelného kódu, který má zefektivnit vývoj aplikací. Knihovna je soubor funkcí, které jsou v případě potřeby volány aplikací. Jsou vytvořené speciálně pro specifický úkol. V porovnání s frameworkem, který funguje jako kostra aplikace. V případě frameworku vývojáři tvoří architekturu aplikace na základě jeho pravidel a struktur. Framework obvykle obsahuje několik knihoven, které jsou v případě potřeby volány. [10]

5. Webová aplikace

Webová aplikace je software, který se používá prostřednictvím webového prohlížeče. Firmy využívají webové aplikace k poskytování vlastních služeb. [11] Webové aplikace slouží jako nástroje pro komunikaci se serverem. Uživatelé mohou prostřednictvím webového prohlížeče odesílat data na server a následně přijímat odpovědi od serveru zpět do svého prohlížeče. [12]

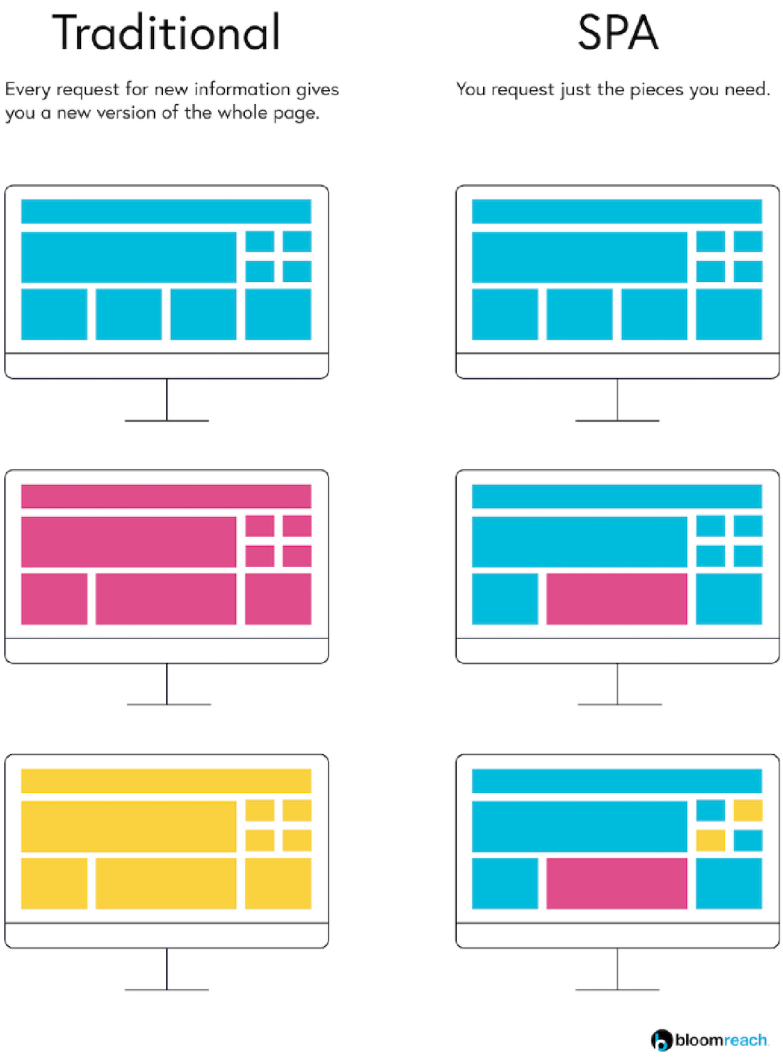
5.1. SPA vs MPA

Společně s webovými aplikacemi se pojí pojem Multi Page Application (dále jako MPA) a Single Page Application (dále jako SPA). Při tvorbě webové aplikace je důležité se rozhodnout, který z těchto dvou modelů bude vhodnější zvolit pro její realizaci.

V případě, že je zapotřebí zobrazit na webové stránce rozsáhlý sortiment produktů či služeb a je zároveň důležité, aby se stránka umísťovala na předních pozicích ve výsledcích vyhledávačů, může být MPA vhodnou volbou. Pokud je klíčovým cílem poskytnout co nejvíce funkcí, či prezentovat rozsáhlá data bez potřeby komunikace se serverem, tak může být vhodnější zvolit SPA. [13]

5.1.1. Single Page Aplikace

SPA umožňuje získat pouze potřebná data ze serveru, která poté prohlížeč zpracuje a vykreslí na webové stránce bez nutnosti opakovaného stažení celé stránky. V MPA se s každým kliknutím na interaktivní prvek odešle požadavek na server, který na základě něho vytvoří HTML dokument, jenž se odešle zpět do prohlížeče uživatele. Důsledkem toho dochází k pomalejšímu načítání obsahu stránky, vyšším nárokům na internetové zdroje a stránka působí méně interaktivně než SPA. V případě SPA není zapotřebí opakovaně ze serveru stahovat hlavičku, patičku, navigační menu a další opakující se elementy stránky. [14]



Obrázek 1. Porovnání fungování MPA a SPA, Zdroj: [14]

SPA odděluje logiku backendu a frontendu, což dovoluje vývojářům pracovat na frontendu nezávisle na zvolené backendové technologii. Pro vývoj SPA lze využít JavaScriptových frameworků jako jsou React, Vue a Angular. [14] Příkladem SPA jsou Trello, Google Disk, Gmail, Facebook, Instagram, Twitter Netflix, Pinterest. [13]

Výhodou SPA je to, že většina zdrojů (HTML, CSS, Scripty) se načtou jednou a poté již není potřeba je znovu načítat. Dochází pouze k přenášení dat. Je jednodušší vytvářet mobilní aplikace, protože vývojáři mohou využít stejný backend kód jak pro webovou aplikaci, tak i pro mobilní aplikaci. SPA se snadněji ladí vzhledem k tomu, že můžeme monitorovat síťové operace, zkoumat jednotlivé elementy stránky a data s nimi spojená. Další výhodou je, že na backendu není zapotřebí implementovat vykreslování HTML stránek, ale pouze poskytování strukturovaných dat (např. JSON a XML). [15]

SPA běžně umí efektivně cachovat lokální data. Data mohou být stažena při prvním spuštění, což umožní uživateli pokračovat v procházení a omezeném používání webové

aplikace i v případě výpadku internetu. V momentě, kdy dojde k obnovení internetového připojení, tak se lokální data synchronizují se serverem. Příkladem je webová aplikace Google Docs. [13]

Nevýhodou SPA je, že mají špatnou optimalizaci pro vyhledávače. SPA funguje na JavaScriptu a umožňuje načítání dat na pozadí bez potřeby obnovování celé stránky pokaždé, když uživatel požaduje nová data. Dochází k tomu, že neexistují samostatné adresy URL optimalizované pro vyhledávače. Řešením tohoto problému je vykreslování na straně serveru. [13]

SPA je náchylné na útoky typu Cross-site skriptování (dále jako XSS). [13] K útokům typu XSS dochází, když útočník využije webovou aplikaci k tomu, aby poslal škodlivý kód ve formě prohlížečového skriptu jinému koncovému uživateli. Prohlížeč koncového uživatele vyhodnotí skript jako důvěryhodný a spustí jej. Je tedy vhodné vždy na serveru zkontrolovat a případně ošetřit data zaslané uživatelem a tím předejít případným škodám. [16]

5.1.2. Multi Page Aplikace

MPA je aplikace složená z více statických stránek, které jsou navzájem propojené odkazy. Při každé změně se musí ze serveru stáhnout znovu celá stránka s veškerými zdroji i přesto, že se některé části stránky mohou opakovat. MPA má obvykle složitou architekturu s více závislostmi. Z důvodu většího provázání frontendu a backendu tyto aplikace často řeší interaktivitu na více vrstvách uživatelského rozhraní. [14]

Příkladem MPA jsou webové stránky typu elektronických obchodů, různých fór a blogů. Příkladem e-shopu mohou být eBay nebo Amazon. Webové stránky tohoto druhu obvykle zpracovávají a předávají rozsáhlé objemy dat mezi serverem a uživatelem, což může vést k dojmu, že nejsou dostatečně interaktivní z pohledu uživatele. [14]

Výhodou MPA je její vysoká škálovatelnost. Počet stránek, které lze přidat do stávající aplikace, není omezen. Když je nutné zobrazit rozsáhlé množství informací, MPA se ukazuje jako vhodnější řešení. [14] Další výhodou je snadná správa optimalizace pro vyhledávače. [15] MPA skvěle fungují s Google Analytics, které vytváří různé užitečné analytické souhrny k jednotlivým stránkám. Díky tomu se dá zjistit, které stránky si vedou

dobře a které ne. Na základě těchto informací lze poté upravit obsah webové aplikace tak, aby došlo ke zvýšení návštěvnosti a času stráveném na ni. [14]

Nevýhodou MPA ovšem může být její komplexnost, protože frontend a backend aplikace je vzájemně svázán. [15] Kvůli neustálému znovu načítání stránky při každé uživatelské interakci, aplikace působí méně interaktivním dojmem. MPA se hůře spravuje, protože je zapotřebí každou jednotlivou stránku aplikace zabezpečit, což může vést k zdlouhavějšímu vývoji. [14]

5.2. Frontend

Je část webové aplikace fungující na straně klienta. Mezi technologie frontendu patří značkovací jazyk HTML, stylovací jazyk CSS a skriptovací jazyk JavaScript.

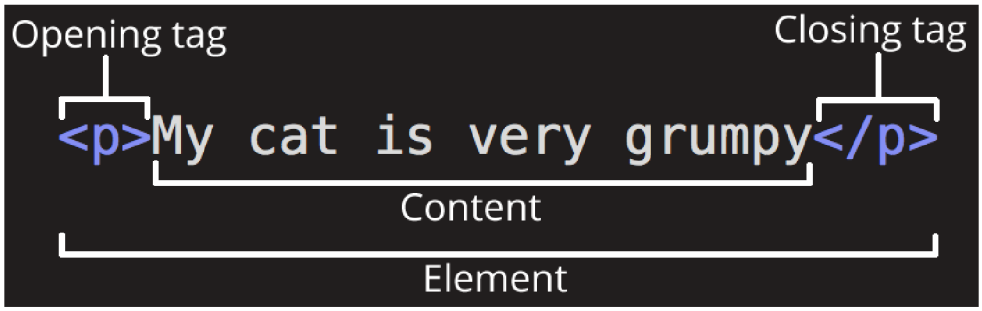
5.2.1. HTML

Hypertextový značkovací jazyk (dále jako HTML) je nejzákladnějším stavebním prvkem webu. Definuje význam a strukturu webového dokumentu. Dalšími stavebními prvky webu jsou potom CSS pro vzhled a JavaScript pro interaktivitu. [17]

Hypertext označuje odkazy, pomocí kterých jsou mezi sebou jednotlivé webové stránky propojeny. Odkazy jsou základním aspektem webu. [17]

HTML definuje strukturu obsahu. Skládá se z řady prvků, které ho obalují, aby došlo k definování jeho účelu a chování. Je-li zapotřebí označit část obsahu jako odstavec, tak ho obalíme do konkrétních tagů. Prvek se může skládat z uvozovacího tagu, který obsahuje jeho název a je ohraničen ostrými závorkami. Tímto způsobem se značí jeho začátek. Za ním následuje jeho obsah, kterým bývá samotný text. Prvek je poté uzavřen jeho názvem, před kterým je nutno ještě připsat lomítko. Název společně s lomítkem je opět ohraničen ostrými závorkami. Jednotlivým prvkům lze poté ještě přiřadit atributy[6], které se vkládají do uvozovacího tagu mezi ostré závorky a za jeho název. Obrázek č. 2 zobrazuje, jak takový prvek vypadá. [17]

[6] **Atribut –** definuje vlastnosti HTML prvku



Obrázek 2. Struktura HTML prvku, Zdroj: [17]

HTML prvky lze také vnořovat, ale je zapotřebí dávat pozor, aby se vždy nejdříve uzavřel vnitřní prvek a poté až vnější. Některé prvky nemají uzavírací tag (jsou nazývány jako nepárové tagy). Příkladem takového tagu je obrázkový tag **<img>**, který neobaluje žádný obsah. Prvek s obrázkem obsahuje atribut se zdrojem k obrázku a atribut s popisem obrázku, který se zobrazí v případě nenalezení cesty k obrázku. [17]

```
<!DOCTYPE html>
<html lang="cs-cz">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Ukázková stránka</title>
  </head>
  <body>
  </body>
</html>
```

Kód 1. Struktura HTML dokumentu, Zdroj: [17]

- **<!DOCTYPE html>** je povinná preambule. Nachází se na začátku HTML dokumentu a zajišťuje, aby prohlížeč dodržel příslušné specifikace. [18]
- **<html></html>** je obalem celé stránky a obsahuje atribut **lang**, který nastavuje primární jazyk dokumentu. [17]
- **<head></head>** je prvek sloužící pro zahrnutí věcí, které nejsou součástí viditelného obsahu stránky. Patří sem klíčová slova, popis stránky, styly a další. [17]
- **<meta charset="utf-8">** je prvek, který slouží pro nastavení znakové sady pro dokument. [17]

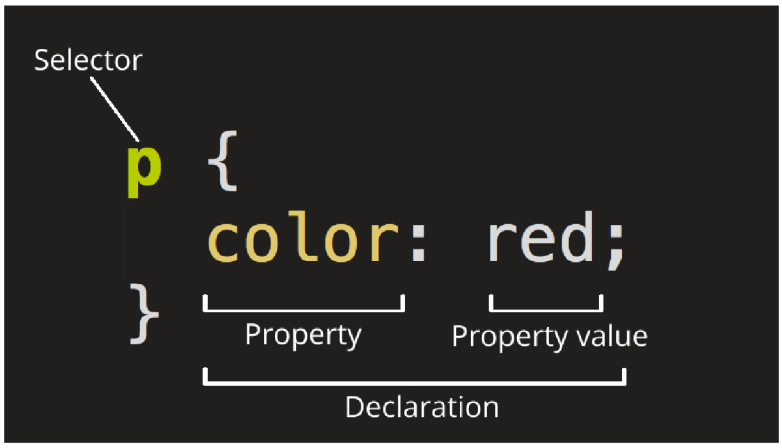
- **<meta name="viewport" content="width=device-width">** zajišťuje vykreslení na šířku **viewportu**[7]. [17]
- **<title></title>** nastavuje název stránky, který se poté zobrazuje v kartě prohlížeče a případně v oblíbených položkách. [17]
- **<body></body>** slouží pro zobrazení veškerého obsahu, který má být dostupný pro uživatele. [17]

HTML obsahuje několik základních prvků pro určení obsahu. Mezi nejzákladnější patří nadpisy. Ty se značí písmenem **h** a mají rozsah od **h1** (největší) do **h6** (nejmenší). Písmeno **p** představuje odstavec, který je dalším důležitým prvkem. Mezi hojně využívané prvky také patří seznamy, které se dělí na dva typy. První z nich je nečíslovaný seznam, ten se značí **ul**, druhým je číslovaný seznam a ten má označení **ol**. Jednotlivé položky seznamu se poté vkládají do **li** tagu. Posledním a asi nejdůležitějším základním prvkem je odkaz značený písmenem **a**. Odkaz má atribut **href**, do kterého se vloží cílová adresa. [17]

[7] **Viewport –** šířka rozložení stránky [19]

5.2.2. CSS

Cascading Style Sheets (dále jako CSS) je stylovací jazyk, který slouží k úpravě vzhledu HTML prvků. CSS je založené na definování sady pravidel. [20]



Obrázek 3. Definice CSS pravidla, Zdroj: [20]

CSS pravidla se skládají z:

- **Selektor** – určuje, které HTML prvky mají být stylovány. [20]
- **Deklarace** – definuje, která z vlastností prvku je stylována. Je vložena do dvou složených závorek, které se nachází za selektorem. [20]
- **Vlastnosti** – jedná se o možnosti, kterými můžeme HTML prvek stylovat. Za každou vlastností se nachází dvojtečka, která ji odděluje od hodnoty. [20]
- **Hodnota** – určuje hodnotu zvolené vlastnosti. Za hodnotou se vždy nachází středník pro oddělení od dalších vlastností. [20]

Lze vybírat i více prvků najednou, pro které budou platit stejná pravidla. Pro zvolení více prvků stačí selektory napsat za sebe a oddělit čárkou. Kromě toho existuje více typů selektorů. Ty lze vybírat na základě názvu prvku. Další možností je přiřadit mu unikátní identifikační číslo (ID). Takový prvek se na stránce může vyskytovat pouze jednou. Je-li zapotřebí aplikovat styly na konkrétní prvky, kterých je na stránce více, lze zvolit selektor s názvem **class**. Prvky lze také vybírat na základě speciálních atributů. Takovým atributem je např. **<img>** [20]

Nejběžnějšími vlastnostmi, které se u prvků dají změnit, je typ písma, jeho velikost a případně řádkování.

CSS rozložení je založené na modelu bloků. Každý blok má důležité vlastnosti a těmi je šířka, odsazení a rámeček. Odsazení se dělí na vnitřní (padding) a vnější (margin). Mezi těmito odsazeními se nachází rámeček (border). [20]

5.2.3. JavaScript

Je programovací jazyk, který umožňuje do webové stránky přidat interaktivitu. JavaScript je relativně kompaktní a flexibilní programovací jazyk, který funguje ve webovém prohlížeči uživatele. [21]

Je-li zapotřebí přidat JavaScriptový skript externě do webové stránky, tak je nutné k němu uvést cestu. Kód č. 2 níže představuje, jak vypadá cesta k externímu JavaScriptovému souboru. [21]

```
<script src="scripts/main.js"></script>
```

Kód 2. Definice externího JavaScriptového souboru v HTML, Zdroj: [21]

JavaScript stejně jako ostatní programovací jazyky obsahuje proměnné, které slouží k ukládání hodnot. Proměnné v JavaScriptu se deklarují pomocí klíčových slov **let** či **var**, za nimiž následuje název proměnné a řádek je ukončený středníkem. Tyto proměnné mohou nabývat různých datových typů. Mezi ně patří String (textový řetězec), Number (číslo), Boolean („pravda/nepravda“), Array (pole) a Object (objekt). [21]

Dalším důležitým prvkem, který JavaScript sdílí s ostatními programovacími jazyky, jsou podmínky. Níže lze vidět příklad zápisu proměnné a podmínky v JavaScriptu. [21]

```
let iceCream = "chocolate";
if (iceCream === "chocolate") {
  alert("Yay, I love chocolate ice cream!");
} else {
  alert("Awwww, but chocolate is my favorite…");
}
```

Kód 3. Ukázka JavaScriptového kódu, Zdroj: [21]

Dalším často využívaným aspektem JavaScriptu jsou funkce, které umožňují zabalit část kódu pro opakovatelné použití. Mezi vestavěné JavaScriptové funkce do webového

prohlížeče patří **document.querySelector**, která vrací specifický html prvek. Další vestavěnou funkcí je **alert**, která zobrazí vyskakovací okno s upozorněním. Lze také definovat své vlastní funkce. [21]

Pro dosažení skutečné interaktivity na internetových stránkách je nezbytné využít události, které odpovídají na uživatelské akce na webu. Nejběžnější takovou událostí je kliknutí na element. Při něm se může provést specifická akce. [21]

5.3. Backend

Je část webové aplikace fungující na straně serveru, který přijímá požadavky od klientů. Za klienta lze označit cokoliv, co posílá požadavky na server. Často se jedná o webové prohlížeče, které server žádají o HTML a JavaScript, aby mohly koncovému uživateli zobrazit příslušnou stránku. Kromě webového prohlížeče může být klientem mobilní aplikace, další služba běžící na jiném serveru nebo chytré zařízení s podporou webu. [22]

5.3.1. Server

Server je počítač, který naslouchá příchozím požadavkům. Jako server se může použít jakýkoliv počítač, který má přístup k internetovému připojení (např. osobní počítač). Tohoto přístupu se využívá při vývoji webových aplikací. Existují však i počítače, které jsou speciálně navrženy a optimalizovány pro serverové aplikace. [22]

5.3.2. Webová aplikace běžící na serveru

Aplikace na serveru implementuje sadu pravidel pro zpracování různých typů požadavků, které jsou určeny pomocí HTTP metod a URL. Spojení HTTP metody a URL se nazývá trasa (route) a jejich přiřazení na základě požadavku se nazývá směrování (routing). Některé z těchto obslužných funkcí budou tzv. middleware. [22] Middleware je software, který funguje jako most mezi různými technologiemi, nástroji a databázemi tak, aby šly snadno integrovat do jednoho systému. [23] Kód, který funguje jako middleware, je spuštěn v době mezi tím, co server přijme požadavek a než odešle odpověď. Funkce middlewaru můžou upravit požadavek, dotazovat se na databázi, nebo jiným způsobem zpracovat příchozí požadavek. Funkce middlewaru obvykle končí tím, že předá řízení další

funkci. Middleware funkce je nakonec volána k ukončení procesu mezi požadavkem a odpovědí, což se projeví odesláním HTTP odpovědi klientovi. [22]

5.3.3. Databáze

Databáze je organizovaný soubor strukturovaných persistentních dat. Existuje vícero druhů databází. Mezi ty nejběžnější patří relační databáze. V nich jsou data uložená ve formě tabulek, které mohou být vzájemně propojené. Pro práci s daty v relačních databázích slouží programovací jazyk Structured Query Language. Ten umožňuje dotazování, manipulaci a definování dat v databázi. Databáze je obvykle řízena systémem pro řízení báze dat (Database Management System). DBMS je databázový software sloužící k vytváření, úpravě a udržování databázových souborů a záznamů. Funguje jako rozhraní mezi databází a jejími koncovými uživateli nebo programy. Má na starost ukládání dat, jejich zálohování, a kromě toho také kontrolu vícero přístupů a zabezpečení. Mezi nejznámější DBMS patří MySQL, Microsoft SQL Server, PostgreSQL a Oracle Database. [24]

5.3.4. Object Relational Mapper ORM

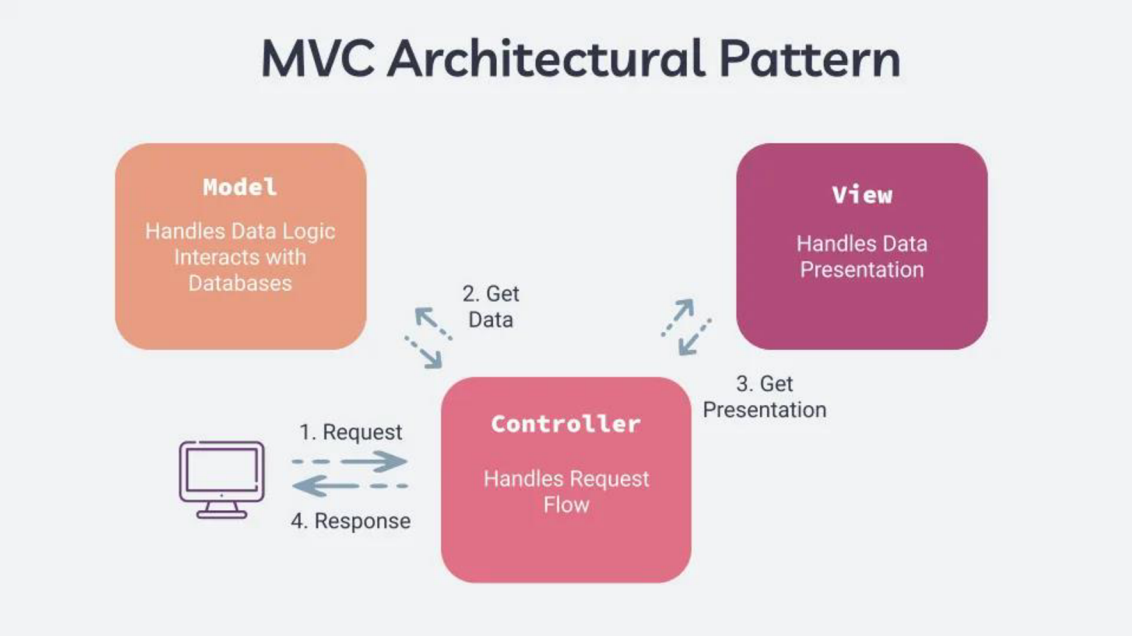
ORM umožňuje definovat třídy a jejich vztahy za pomocí programovacího jazyka např. Pythonu bez toho, aniž by bylo zapotřebí znát využívanou databázi. Framework Django obsahuje integrovaný ORM systém, který dokáže převést objektově orientovaný kód napsaný v Pythonu do skutečné databáze obsahující tabulky, definice dat a všechny možné operace související s databází. Tento přístup poskytuje rychlejší vývoj a jednodušší údržbu aplikace. V případě frameworku Django stačí změnit v nastavení projektu využívanou databázi a poté provést migraci modelů bez toho, aniž by bylo zapotřebí napsat jakýkoliv SQL kód. Na základě modelů se poté automaticky pro danou databázi vygeneruje příslušný kód. [25] Nejznámější ORM pro programovací jazyk Python jsou Django ORM a SQLAlchemy. [26]

5.3.5. Architektonické vzory

Architektonické vzory definují strukturu a organizaci softwarového systému. Vymezují základní komponenty, jejich interakce a celkové uspořádání daného systému. Určují škálovatelnost, výkonnost a udržitelnost systému. [27]

5.3.5.1. Model View Controller (MVC)

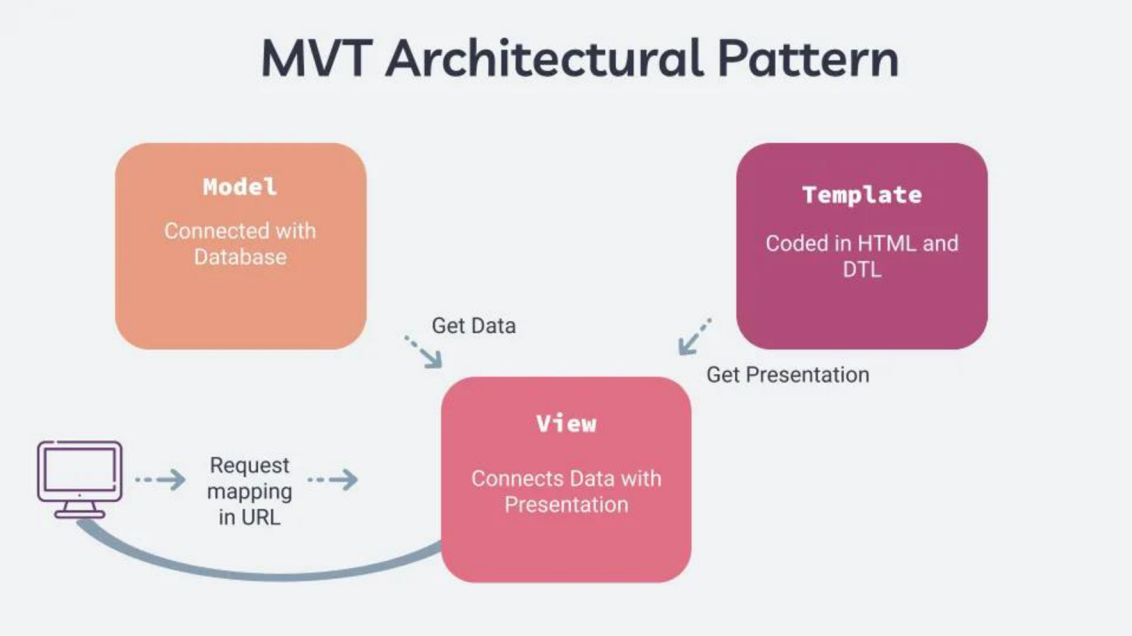
Návrhový vzor MVC se rozděluje na tři části. První je model, který má za úkol zpracovávat data a logiku. Komunikuje s databází a získává data podle požadavku. Druhou částí je view, který se stará o to, jak by měla být data reprezentována. Poslední částí je kontrolér řídící tok požadavků. Funguje jako prostředník, který přijímá požadavky a odesílá odpovědi. Zároveň během tohoto procesu komunikuje s modelem a s view. Když přijde požadavek, tak je předán zvolenému kontroléru, aby ho zpracoval. Potřebná data jsou získána z modelu a jsou upravená do vhodné prezentace pomocí view a následně je kontrolér odešle zpět klientovi. [28]



Obrázek 4. Architektura MVC, Zdroj: [28]

5.3.5.2. Model View Template (MVT)

Návrhový vzor MVT taktéž rozděluje aplikaci na tři části. První částí je model, který funguje obdobně jako v případě MVC. Má na starost data, logiku a zároveň komunikuje s databází. Druhou částí je view, který funguje odlišným způsobem než v případě MVC. Zde view přijímá požadavky a vrací odpovědi. Poslední částí je template, který slouží jako prezentační vrstva běžně napsaná v HTML nebo v Django Template Language. V případě příchozího požadavku Django projde požadavek různými vzory URL, aby ho přiřadilo ke správnému view. Ten se pak zkoordinuje s modelem a s template, aby mohl klientovi odeslat zpracovanou odpověď. [28] Výhodou může být to, že v případě MVC jsou obvykle všechny tři části napsané ve stejném jazyce. V případě frameworku Django, který využívá MVT, tomu tak není, model a view jsou napsané v Pythonu a template poté v HTML. To může znamenat výhodu při vývoji, kde se vývojáři programující v Pythonu mohou soustředit na model a view, mezitím co specialista na HTML může pracovat na template. [25]



Obrázek 5. Architektura MVT, Zdroj: [28]

6. Představení frameworků

Následující kapitola se zabývá základním popisem jednotlivých frameworků, které budou později porovnávány. Kapitola začne představením velice populárního frameworku Django. Dále se zaměří na frameworky Flask, Tornado, CherryPy, Pyramid a nakonec Bottle.

6.1. Django

Django je vysokoúrovňový framework pro tvorbu webových aplikací v Pythonu. [29] Za jeho vznikem stojí dva lidé. Jsou jimi Adrian Holovaty a Simon Willison, kteří na něm začali pracovat v roce 2003. V roce 2005 bylo Django oficiálně vydáno a pojmenováno po belgicko-francouzském kytaristovi Django Reinhardtovi. [30] Poskytuje velké množství nástrojů a funkcí, které zjednodušují proces vývoje aplikace. Vede vývojáře k využívání již existujících komponent bez potřeby jejich opětovného vytváření. Django nabízí základní služby jako je směrování, šablonování, autentizace, migrace databáze a testování. Framework poskytuje téměř vše, co by mělo být zapotřebí pro vývoj webové aplikace. Výhodou Djanga je také jeho univerzálnost. Lze ho využít k tvorbě jakéhokoli typu webové stránky. Od jednoduchých statických stránek, přes dynamické stránky s obsahem, až po komplexní webové aplikace s pokročilými funkcemi. Příkladem mohou být sociální sítě, elektronické obchody, blogy a další. Django má v sobě vestavěnou bezpečnostní ochranu vůči běžným hrozbám. Django obsahuje integrovaný systém ORM, známý jako Django ORM, který slouží pro interakci s databází. Pro tvorbu webových aplikací využívá architektonický vzor MVT. [29]

6.2. Flask

Flask je minimalistický framework pro tvorbu webových aplikací v Pythonu. Je navržený tak, aby nabídl základní funkce, ale v případě potřeby umožňoval přidání dalších funkčností. Flask obsahuje tři hlavní závislosti. Werkzeug, což je sada nástrojů pro WSGI. [31] To je rozhraní mezi webovým serverem a webovými aplikacemi. Popisuje, jak s nimi webový server komunikuje a jak mohou být vzájemně propojeny pro zpracování jednoho požadavku. [32] Další závislostí je Jinja2, což je šablonovací systém, který usnadňuje tvorbu HTML šablon. Poslední závislostí je Click, který nabízí integraci příkazového řádku. Flask nemá vestavěnou podporu pro databáze, validaci formulářů, autentizaci

uživatelů a ani pro jiné složitější úlohy. Tyto a další důležité služby pro webové aplikace jsou dostupné prostřednictvím rozšíření, která se integrují s jádrem. Vývojář má možnost si vybrat rozšíření, která mu nejvíce vyhovují, nebo si vytvořit vlastní. [31]

6.2.1. Dash

Open source framework Dash slouží pro vytváření webových aplikací. Je založený na frameworku Flask, který byl popsán v předchozí podkapitole. Vznikl v roce 2017 a uplatnění nachází především pro rychlé vytváření webových aplikací, které se zaměřují na analýzu dat a jejich zobrazení. Dash spojuje tři technologie. První z nich je Flask, který obstarává funkčnost webového serveru. Druhou je poté React.js, který je využit pro tvorbu uživatelského rozhraní. Poslední technologií je Plotly.js. Jde o knihovnu sloužící pro generování grafů, jež jsou implementovány do webové aplikace. [33]

6.3. Tornado

Tornado je webový framework a asynchronní síťová knihovna pro programovací jazyk Python. Je dobře škálovatelný. Díky tomu umožňuje škálovat až na desítky tisíc otevřených spojení. A to za pomoci použití neblokujícího síťového vstupu a výstupu (I/O). Je ideální pro aplikace, které vyžadují dlouhodobé spojení mezi uživatelem a serverem. Framework Tornado není založen na WSGI. [34] Obvykle běží pouze jedno vlákno[8] pro jeden proces. [34]

6.4. CherryPy

CherryPy je minimalistický objektově orientovaný webový framework pro programovací jazyk Python. Stejným způsobem, jakým by se vytvářel objektově orientovaný program v jazyce Python, se zde tvoří webová aplikace. To může vést k přehlednějšímu zdrojovému kódu. Už více než deset let se CherryPy využívá pro mnohé weby od těch jednodušších ke složitějším. (např. gutenberg.org) [35] Jeho první verze byla vydána v červnu roku 2002 francouzským hackerem Remi Delonem. [36] Oproti ostatním známým webovým frameworkům pro Python neposkytuje vestavěnou podporu

[8] **Vlákno –** jednotka umožňující paralelní zpracování úloh v rámci jednoho procesu

pro vícevrstvou architekturu a ani nástroje pro práci s frontendem. CherryPy svěřuje tuto odpovědnost do rukou vývojářů webových aplikací. Je navržený tak, aby měl co nejsrozumitelnější rozhraní, a i přesto poskytoval spolehlivou kostru pro tvorbu webové aplikace. [37]

6.5. Pyramid

Webový framework Pyramid je navržený pro snadnější tvorbu webových aplikací. Je distribuován jako open source. Pyramid je konstruován s cílem, aby byl vývoj v něm intuitivní. Poskytuje základní nástroje potřebné pro webové aplikace jako je mapování URL na kód a zabezpečení. Kromě toho poskytuje i další užitečné nástroje jako je šablonování a integrace s databázemi. Pyramid je navržen pro rychlost a je pečlivě testován s cílem minimalizovat počet chyb. Je plně kompatibilní s verzí 3 programovacího jazyka Python. Díky tomu je možné využít jeho nových funkcí. Velkou výhodou Pyramid je jeho škálovatelnost. Umožňuje snadné vytvoření malé webové aplikace a její následné rozšíření do větších rozměrů. [38]

6.6. Bottle

Bottle je typ WSGI mikro webového frameworku pro programovací jazyk Python. [39] Bottle se vyznačuje rychlostí, jednoduchostí a nízkou náročností na prostředky. Je distribuován jako jediný soubor modulu, který nevyžaduje žádné další knihovny kromě těch, které jsou součástí standardní knihovny Pythonu. [39]

Pro vývoj webových aplikací poskytuje Bottle vývojářům několik funkcí. Bottle umožňuje definovat, jaké funkce se mají volat v závislosti na URL požadavku, a podporuje čisté a dynamické URL. Dále nabízí svůj rychlý vestavěný šablonovací systém. Podporuje také jiné šablonovací systémy, jako jsou Jinja2, Mako nebo Cheetah. Kromě toho poskytuje též nástroje pro snadnou práci s formulářovými daty, nahráváním souborů, cookies a hlavičkami. [39]

7. Metodiky porovnávání

Předchozí kapitola se zabývala představením jednotlivých frameworků. V nadcházející kapitole budou uvedeny metodiky, na základě kterých budou tyto frameworky porovnávány.

7.1. Dokumentace

Ačkoli to nemusí být na první pohled zřejmé, dokumentace může být důležitým faktorem ovlivňujícím rychlost vývoje aplikace v daném frameworku. Dobře zdokumentovaný framework dokáže vývojářům ušetřit spoustu času a úsilí při práci. Pokud lze v dokumentaci rychle najít odpovědi na otázky a řešení problémů, tak se tím výrazně zefektivní vývoj aplikace. Dokumentace obsahující příklady kódů a návody, které ukazují, jak daný framework používat v praxi, poskytne vývojářům lepší pochopení jeho funkcí a možností. Pro dlouhodobou udržitelnost projektu v daném frameworku je důležité, aby dokumentace obsahovala informace ohledně jeho vývoje a provedených změn v jednotlivých verzích. Celkově lze říci, že kvalitní dokumentace umožňuje lépe porozumět danému frameworku, efektivněji s ním pracovat a snadněji se rozhodovat při jeho výběru.

7.2. Zabezpečení

Webové aplikace jsou často cílem útoků, použitím frameworku s dobrou bezpečnostní podporou lze snížit riziko zranitelnosti, úniku dat, neoprávněného přístupu a dalších hrozeb. Kvalitní bezpečnostní funkce zabudované v rámci frameworku umožňují řešit problémy efektivní cestou a snižují potřebu hledání vlastního řešení těchto problémů. Frameworky, které mají integrované bezpečnostní funkce, navíc mnohdy dodržují osvědčené postupy a standardy, což může usnadnit vývoj bezpečných aplikací.

7.3. Aktualizace

Lze říci, že pravidelnost aktualizací je dalším klíčovým faktorem pro porovnávání frameworků z několika důvodů. Pravidelné aktualizace zajišťují, že je framework chráněn před nově vznikajícími bezpečnostními hrozbami. Aktualizace často opravují chyby a zlepšují stabilitu frameworku. Kromě toho přidávají nové funkce a vylepšení. Pravidelně se rozvíjející framework může lépe reagovat na nové trendy a potřeby vývojářů. Dále pak udržuje krok s nejnovějšími technologiemi, na kterých je postavený, což zaručuje dlouhodobou udržitelnost projektu. Frameworky, které jsou pravidelně aktualizované, mají mnohdy aktivnější komunitu a lepší podporu. Vývojáři tedy mohou rychleji řešit problémy, což může vést k efektivnějšímu vývoji aplikace.

7.4. Sociální metriky

Sociální metriky mohou být též klíčovým faktorem pro porovnání jednotlivých frameworků. To, jak je framework oblíbený na GitHubu a jeho počet stažení, dokáže poukázat na to, jak velkou komunitu má. Ta je klíčová pro vývoj samotného frameworku a pro vývoj aplikací v něm. Větší množství lidí může přispět k řešení problémů, k hledání chyb ve frameworku, a i s jeho samotným vývojem. Kromě toho komunita často vytváří a sdílí užitečné knihovny, které mohou pomoci při práci v daném frameworku.

7.5. Výkon frameworků

To, jak je framework efektivní a jak dokáže obstát při větší zátěži je jedním z nejdůležitějších faktorů. Ten, který je nestabilní a pomalý, nemusí být vhodným kandidátem pro vývoj aplikací.

Všechny frameworky budou testovány za stejných podmínek a s využitím softwaru K6. Jedná se o open-source nástroj pro testování výkonů systému. Testováno bude na hardwaru s následujícími specifikacemi:

- **OS:** Windows 11 x64
- **CPU:** Intel(R) Core(TM) i7-5820K CPU @ 3.30GHz
- **RAM:** 32 GB 2133 MHz
- **GPU:** AMD Radeon RX 6750XT 12 GB

8. Porovnání jednotlivých frameworků

V této kapitole budou porovnány jednotlivé frameworky na základě metodik představených v předchozí kapitole.

8.1. Dokumentace

Dobře zdokumentovaný framework je klíčový pro efektivní vývoj. Kvalitní dokumentace ušetří čas a usnadní práci vývojářům. V této podkapitole tedy bude popsána dokumentace jednotlivých frameworků.

8.1.1. Django

Při otevření Django dokumentace si lze všimnout, že je rozdělená do několika sekcí. Úvodní kapitola dokumentace se zabývá prvními kroky práce s Djangem. Obsahuje kvalitně zpracovaný návod, který začínající Django vývojáře provede krok za krokem k vytvoření jejich první aplikace. Návod je strukturovaný do několika částí, přičemž každá z nich se zaměřuje na specifickou oblast, která je klíčová pro vývoj aplikace. Vše v návodu je stručně, ale dostatečným a pochopitelným způsobem vysvětleno. Lze říci, že návod dá začínajícímu vývojáři velice kvalitní podklad pro tvoření budoucích aplikací. Další sekce na úvodní stránce dokumentace se zabývá tím, jakým způsobem je organizovaná. Obsahuje čtyři důležité odkazy. První z nich „Tutorials“ navede nováčky na návod k instalaci frameworku a k vytvoření první aplikace. Dalším odkazem je „Topic guides“. Pod ním jsou obsažena klíčová témata a koncepty popsané na detailní úrovni. Třetím odkazem je „Reference guides“, pod nímž se nachází technické reference pro APIs. Posledním odkazem je poté „How-to guides,“ který obsahuje užitečné instrukce na různé typy otázek typu „Jak“. Na úvodní stránce dokumentace jsou rovněž další sekce, které shrnují témata s podobnou tematikou. [40]

Velice užitečným prvkem v dokumentaci je možnost si přepnout mezi jednotlivými verzemi frameworku Django. Tato možnost se skrývá v pravém dolním rohu dokumentace pod názvem: „Documentation version: x,“ po najetí myši se zobrazí jednotlivé verze, které lze měnit. Na výběr je velké množství verzí. Od verze 1.8 až po aktuální stabilní verzi 5.0 nebo momentálně vyvíjenou verzi. Nad možností přepínání mezi verzemi se nachází další užitečný prvek a tím je možnost si změnit jazyk dokumentace. Kromě angličtiny si lze

vybrat z několika jazyků, mezi které patří např. francouzština nebo polština. Dokumentace bohužel zatím postrádá český jazyk. Pro některé nabízené jazyky jako je např. řečtina, je dokumentace momentálně přeložena pouze z části. [40]

Vývojáři jistě ocení i možnost si dokumentaci stáhnout v offline podobě. Na výběr je ze tří formátů: HTML, PDF a ePub. Celkově je dokumentace velice rozsáhlá, organizovaná, dobře popsaná a lze v ní velice rychle hledat specifické téma. [40]

8.1.2. Flask

Hned na úvodní stránce dokumentace si lze všimnout, že je strukturovaná do třech sekcí. První z nich je uživatelská příručka, která obsahuje důležité kategorie zabývající se jednotlivými částmi vývoje webové aplikace. Každá z nich má pod sebou jednotlivá témata, která se k ní váží. Při rozkliknutí kteréhokoli tématu je uživatel přesměrován na stránku konkrétní kategorie, která obsahuje všechna pod ní spadající témata, a zde je přesunut na místo daného tématu. Tento přístup je pro uživatele velice příjemný, protože není zapotřebí se při hledání dalšího tématu spjatého s danou kategorii proklikávat jedním či více odkazy, ale stačí se podle potřeby posunout na stránce nahoru či dolů. Jednotlivá témata jsou velice kvalitně a srozumitelně zpracována a není tedy zapotřebí se odkazovat na externí zdroje pro pochopení daného tématu. Uživatelská příručka taktéž obsahuje i návod na tvorbu vzorové blogovací aplikace, která začínající vývojáře postupně provede pochopitelným způsobem jednotlivými kroky, které jsou zapotřebí pro vytvoření plnohodnotné webové aplikace. Další sekcí dokumentace je API reference, která obsahuje informace k specifickým funkcím, třídám a metodám. Poslední se poté zabývá dodatečnými poznámkami, které popisují jednotlivá návrhová rozhodnutí a popis toho, jak se tvoří dodatečné balíčky do Flasku. Dodatečné poznámky také obsahují informace k jednotlivým verzím frameworku. [41]

Flask dokumentace též nabízí možnost si skrze plovoucí tlačítko nacházející se v pravém dolním rohu stránky, změnit její verzi a jazyk. Ačkoli je Flask oblíbeným frameworkem v Pythonu, tak i přesto neposkytuje možnost stáhnout si dokumentaci pro offline použití. Celkově Flask dokumentace působí organizovaně a je otevřená nově začínajícím vývojářům, kteří nemají o webových aplikacích až takový přehled. [41]

8.1.3. Tornado

Úvodní stránka dokumentace pro Tornado zahrnuje stručný popis frameworku a implementaci nejjednodušší příkladové aplikace „Hello world“. Na levé straně dokumentace se nachází navigační menu obsahující důležité odkazy. Prvním z nich je uživatelská příručka, která zahrnuje témata, zabývající se důležitými částmi potřebnými k vývoji webové aplikace. Jednotlivá témata jsou stručně, ale srozumitelně popsána. Další sekcí nacházející se v navigačním menu je „Web framework“. Ta obsahuje informace o třídách a jejich metodách, které jsou klíčové pro vývoj aplikací v Tornadu. Dále se v navigačním menu nachází odkazy zabývající se např. asynchronním síťováním, HTTP servery a klienty. I když dokumentace usilovně seskupuje související témata, někdy je nutné pro navigaci k určité oblasti kliknout na několik odkazů, což může zvyšovat nepřehlednost dokumentace. Ta sice obsahuje vyhledávání, které tento problém částečně řeší, ovšem začínající vývojáři, kteří ještě neznají všechny třídy a jejich metody, se musí do hloubky proklikávat dokumentací. S tím souvisí to, že součástí Tornado dokumentace není návod se vzorovou aplikací, který by provedl začínající vývojáře krok za krokem k vytvoření plnohodnotné webové aplikace. [42]

Tornado nabízí svou offline verzi dokumentace ve dvou formátech. Prvním z nich je PDF a druhým je Epub[9]. Tato možnost stažení pro offline použití se nachází v levé spodní části společně s možností přepnutí mezi verzemi. Chybí zde možnost si změnit jazyk. Dokumentace je dostupná pouze v anglickém jazyce. Celkově je dokumentace stručná, ale vystihující. Organizace je pochopitelná, ale některým začínajícím vývojářům by mohla připadat málo přehledná. [42]

8.1.4. CherryPy

Již z prvního pohledu na dokumentaci je zřejmé, že CherryPy je navrženo pro jednoduchý a přívětivý přístup pro vývojáře seznamující se s vývojem webových aplikací. Úvodní stránka dokumentace poskytne stručný přehled frameworku a ukáže kód pro nejjednodušší aplikaci „Hello World“. Navigační menu je umístěno na levé straně a je organizováno do několika oddílů. První s názvem „Předmluva“ se zabývá důvody, proč použít pro tvorbu webových aplikací právě CherryPy a jaké populární služby ho využívají. V navigačním menu se samozřejmě nachází sekce věnovaná instalaci frameworku.

[9] **Epub –** formát souboru pro elektronické knihy

Kromě toho zde lze najít také sekci nazvanou „Tutorials“. Pod ní se nachází návod, který provede začínající vývojáře krok za krokem frameworkem a představí mu jeho základní koncepty, ale i několik těch pokročilejších. Asi nejdůležitější kapitoly jsou „Basics“ a „Advanced“. „Basics“ zahrnuje základní koncepty webové aplikace v CherryPy. Vše je tu srozumitelně popsáno a nechybí zde ani zvýrazněné upozornění na podstatná fakta. Sekce „Advanced“ se zabývá pokročilejšími funkcemi frameworku. Stejně jako v případě předchozí kapitoly je zde vše srozumitelně popsáno a nachází se tu i obrázky pro lepší pochopení daného tématu. [35]

Dokumentace má implementovanou funkci pro změnu verze, jenže se zde nachází pouze jediná volba a tou je aktuální verze. CherryPy bohužel nenabízí offline verzi své dokumentace. To může být pro některé vývojáře omezující. Jinak je dokumentace celkově velice dobře strukturovaná, přehledná a jednotlivá témata jsou kvalitně popsána tak, aby je dokázali pochopit i méně zdatní vývojáři. [35]

8.1.5. Pyramid

Na úvodní stránce dokumentace frameworku Pyramid lze najít příklad aplikace „Hello world“ následovaný několika dalšími sekcemi. Jedna z nich se nazývá „Tutorials“ a jak už název napovídá obsahuje návody, pro práci s Pyramid frameworkem a jeho komponentami. Jeden z návodů je určen začínajícím vývojářům, aby jim pomohl vytvořit plnohodnotnou webovou aplikaci. Tento návod je poměrně obsáhlý a rozdělený do několika desítek menších kapitol. Každá z nich představuje nějakou podstatnou část frameworku. Samotné kapitoly jsou poté opravdu velice stručně popsané. Většinu stránky pokrývá kód bez hlubšího vysvětlení a až na konci stránky se nachází několika řádková analýza toho, co se v kapitole přidalo. Vývojář tak musí neustále přecházet mezi sekcemi s kódem a analýzou. Tou nejdůležitější sekcí v dokumentaci je poté „Narrative Documentation“. Zde jsou srozumitelně organizovaná jednotlivá témata potřebná pro vývoj webové aplikace. Každé jednotlivé téma je po rozkliknutí podrobně a výstižně popsáno. Poměrně nepříjemný z uživatelského hlediska může být způsob navigace v dokumentaci. V levé části stránky se nachází navigační menu sloužící pouze k navigaci v rámci právě otevřeného tématu. Pro změnu tématu je vždy nutné přejít na úvodní stránku a přesunout se dolů. Je to velice nepraktický přístup. Další důležitou sekcí je dokumentace API, která slouží jako referenční materiál pro veřejné API poskytované Pyramidem. [43]

Offline dokumentaci frameworku Pyramid si lze stáhnout přes plovoucí tlačítko v pravém dolním rohu webové stránky. Na výběr je ze dvou formátů: PDF a Epub. Dokumentace je dostupná pouze v anglickém jazyce. Kromě nepraktické navigace a hodně stručnému návodu pro začátečníky je jinak celkově dokumentace organizovaná a jednotlivá témata jsou dobře pochopitelná. [43]

8.1.6. Bottle

Po otevření dokumentace k frameworku Bottle je zřejmé, že je navržena s důrazem na jednoduchost a přehlednost. Na úvodní stránce se nachází stručný přehled frameworku spolu s demonstrační aplikací „Hello World“. Níže na stránce se poté nachází jednotlivé sekce. Tou nejvýznamnější je uživatelská příručka. Ta obsahuje pouze několik kapitol. Jednou z nich a velice důležitou je návod, který zahrnuje potřebné informace k vývoji webové aplikace v tomto frameworku. Návod je velice podrobně a srozumitelně popsán a slouží jako hlavní bod, od kterého se má vývojář odrazit. Poté se zde nachází pár dalších kapitol zabývajících se náležitostmi spjatými s frameworkem, jako je např. konfigurace aplikace. Kromě toho dokumentace Bottle obsahuje sekci „Knowledge base“, která zahrnuje kolekci užitečných článků a návodů. [44]

Dokumentace frameworku Bottle nabízí možnost stažení v offline formě ve formátu PDF nebo HTML a je dostupná pouze v anglickém jazyce. Dokumentace je opravdu stručná a nenabízí toho mnoho a navigace v dokumentaci zde funguje velice podobným způsobem jako v případě frameworku Pyramid, což je poměrně nepraktické. [44]

8.2. Zabezpečení

Následující podkapitola se zabývá zabudovanými bezpečnostními funkcemi v konkrétních frameworcích. Některé z nich budou vybrány a stručně popsány.

8.2.1. Django

Django je jedním z frameworků, který je vybaven velkým množstvím vestavěných bezpečnostních funkcí. Některé z nich jsou představeny níže.

8.2.1.1. Cross-Site Scripting (XSS)

XSS útoky umožňují uživatelům vkládat skripty na straně klienta do prohlížečů jiných uživatelů. Tento typ útoku je často realizován tím, že škodlivé skripty jsou uloženy v databázi a následně načteny a zobrazeny ostatním uživatelům. Alternativně může útočník přimět uživatele kliknout na odkaz, který spustí jeho vlastní JavaScriptový kód v prohlížeči oběti. XSS útoky mohou pocházet z jakéhokoli nedůvěryhodného zdroje dat, např. z formulářů, cookies a webových služeb. Pro minimalizaci rizika je tedy nezbytné provést důkladné očištění dat před jejich vložením do webové stránky. [45]

Webová aplikace využívající šablon Django je chráněná před většinou útoků typu XSS. Django šablony dokáží speciální znaky (např. HTML tagy) převést na jejich bezpečnou reprezentaci tak, aby nemohly být interpretovány jako kód. V případě potřeby je možné tuto automatickou záměnu znaků vypnout. Lze ji vypnout pro specifické proměnné, či pro celé bloky kódu. Ovšem využívání funkce automatické záměny v Django šablonách nezaručí stoprocentní bezpečnost před XSS útoky. I zde existují způsoby, jak tento systém obelstít. Je zásadní být opatrný při ukládání HTML kódu do databáze, zejména když má být tento kód následně vytažen z databáze a prezentován uživateli. Django šablony umí zpracovat HTML, ale pokud se má koncovému uživateli zobrazit obsah v jiném formátu, je zapotřebí pro tento formát zajistit vlastní implementaci záměny znaků. [45]

8.2.1.2. Cross-Site Request Forgery (CSRF)

CSRF útoky umožňují útočníkovi provádět akce pomocí přihlašovacích údajů jiného uživatele bez toho, aniž by o tom dotyčný uživatel věděl nebo s tím souhlasil. [45]

Django má vestavěnou ochranu proti většině útoků typu CSRF. Ochrana je ve výchozím stavu aktivovaná. V případě potřeby lze ochranu vypnout a využít ji pouze na specifických místech. Ovšem tento přístup se nedoporučuje. Kromě toho jsou zde určitá omezení, pokud má vaše webová stránka subdomény, které nespadají pod vaši kontrolu. [45]

Django ochrana proti CSRF funguje na tom principu, že v každém POST požadavku kontroluje tajný klíč. To zajistí, že útočník nemůže simulovat odeslání formuláře na daném webu jako jiný uživatel. K tomu, aby to mohl útočník provést, tak by musel znát tajný klíč, který je pro každého uživatele specifický. Pokud je daná webová

stránka nasazena s HTTPS, tak se middleware CsrfViewMiddleware ujistí, že odkazující URL pochází z téže domény a portu, což zvyšuje bezpečnost proti CSRF útokům. [45]

8.2.1.3. SQL Injection

SQL injection je útok, při kterém je útočník schopný spustit libovolný SQL kód na databázi. To může vést např. k mazání záznamů nebo k úniku dat. [45]

Django své dotazy chrání před SQL injection tím, že jsou parametrizovány. To znamená, že SQL kód dotazu je oddělen od jeho parametrů. Ty, které může uživatel poskytnout a které by mohly představovat potenciální riziko, jsou bezpečně zpracovány databázovým ovladačem. Django také umožňuje vývojářům psát přímé dotazy nebo provádět vlastní SQL. V tom případě je ale nutné manuálně zabezpečit všechny parametry. [45]

8.2.1.4. Clickjacking ochrana

Clickjacking je útok, při kterém webová stránka obaluje jiný web do rámce. To znamená, že na první pohled uživatel vidí normální stránku, ale ve skutečnosti je tato stránka zobrazena uvnitř rámce jiné stránky. Cílem útoku je oklamat uživatele tak, aby prováděl akce na cílovém webu bez svého vědomí. [45]

Django obsahuje ochranu proti clickjackingu pomocí middlewaru X-Frame-Options. Tento middleware umožňuje omezit zobrazení webu uvnitř rámce v podporovaných prohlížečích. Je možné tuto ochranu zakázat na úrovni pohledu, nebo konfigurovat přesnou hodnotu hlavičky, pokud je to potřeba. Tento middleware je vhodný pro jakýkoli web, který nepotřebuje, aby jeho stránky byly zobrazeny v rámci na webech třetích stran. [45]

8.2.2. Flask

Stejně jako v případě Djanga i vývojáři Flasku se snaží implementovat bezpečnostní funkce proti nejznámějším typům útoků.

8.2.2.1. Cross-Site Scripting (XSS)

Ve Flasku je šablonovací systém Jinja2 automaticky nakonfigurován tak, aby převedl všechny speciální znaky v šablonách na jejich bezpečnou reprezentaci. Toto opatření minimalizuje riziko XSS při využívání šablon. Tato ochrana je však dostupná pouze v případě využívání Jinja2. Pokud je HTML obsah generován bez jeho využití, je zapotřebí zavést vlastní implementaci zabezpečení. Jinja2 nechrání před útoky, které zahrnují vkládání atributů, proto je důležité vždy atributy obalit do uvozovek. Kromě toho zde není ochrana před posíláním HTML, nebo jiného textového souboru z nahrávaných souborů. [46]

8.2.2.2. Cross-Site Request Forgery (CSRF)

Flask neprovádí automatickou ochranu proti CSRF útokům z několika důvodů. Jedním z nich je, že ideálním místem pro tuto ochranu by byl framework pro validaci formulářů. Ve Flasku však žádný takový neexistuje. Je tedy zapotřebí tuto ochranu implementovat svým vlastním způsobem, nebo využít externí framework. Způsob, kterým by se tato ochrana dala implementovat, by mohl být následující. Každý požadavek, který modifikuje obsah na serveru, by měl svůj jednorázový token uložený v cookie a odesílal by ho spolu s daty formuláře. Po obdržení dat na serveru by muselo dojít k ověření, že jsou oba tokeny shodné. [46]

8.2.2.3. Clickjacking ochrana

Flask má stejně jako Django zabudovanou clickjacking ochranu. [46] Ta má dvě možnosti, mezi kterými lze zvolit. První je 'DENY', ta funguje tak, že stránka nemůže být zobrazena v rámci bez ohledu na to, z jakého webu je načítána. Druhou možností je 'SAMEORIGIN', ta znamená to, že stránka může být zobrazena pouze tehdy, pokud všechny nadřazené rámce pochází ze stejného zdroje jako samotná stránka. [47] Způsob, kterým lze tuto ochranu ve Flasku nastavit, je ukázán níže. [46]

```
response.headers['X-Frame-Options'] = 'SAMEORIGIN'
```

Kód 4. Ukázka nastavení clickjacking ochrany ve Flasku, Zdroj: [46]

8.2.3. Tornado

Tornado se neustále rozvíjí a zavádí nové ochranné funkce před nebezpečnými útoky, aby nezůstal pozadu oproti ostatním frameworkům.

8.2.3.1. Cross-Site Request Forgery (CSRF)

CSRF je běžným problémem u personalizovaných webových aplikací, tudíž stejně jako v případě Djanga a Flasku i Tornado má proti tomuto typu útoku vestavěnou ochranu. Jak tuto ochranu nastavit speciálně pro framework Tornado je zobrazeno níže. [48]

```
settings = {
    "cookie_secret": "RANDOM_GENERATED_VALUE",
    "login_url": "/login",
    "xsrf_cookies": True,
}
application = tornado.web.Application([
    (r"/", MainHandler),
    (r"/login", LoginHandler),
], **settings)
```

Kód 5. Ukázka nastavení CSRF ochrany v Tornadu, Zdroj: [48]

8.2.3.2. DNS Rebinding

DNS rebinding je typ útoku, který umožňuje externím webům přístup k prostředkům na soukromých sítích. Útok spočívá v použití DNS jména, které střídavě vrací IP adresu ovládanou útočníkem a IP adresu ovládanou obětí. Tento útok může umožnit útočníkovi získat přístup k citlivým informacím na soukromých sítích, ke kterým by jinak neměl přístup. Je tedy důležité, aby aplikace byly chráněny proti DNS rebindingu. Toho lze dosáhnout např. validací HTTP hlavičky „Host“. Tornado má proti tomuto typu útoku vestavěnou ochranu. Jak může vypadat implementace validace HTTP hlavičky 'Host' v Tornadu, lze vidět níže. [48]

```
app = Application([
    (HostMatches(r'(localhost|127\.0\.0\.1)'),
        [('/foo', FooHandler)]),
    ])
```

Kód 6. Ochrana proti DNS rebindingu v Tornadu, Zdroj: [48]

8.2.4. CherryPy

CherryPy nabízí pouze velice omezenou vestavěnou sadu bezpečnostních funkcí. Poskytuje základní ochranu proti útokům typu Clickjacking a XSS. Bohužel CherryPy nenabízí CSRF ochranu. Je tedy nutné využít externích knihoven, nebo je zapotřebí ji implementovat svým vlastním způsobem. [49]

8.2.5. Pyramid

Pyramid nabízí důležité bezpečnostní ochrany. Při vývoji je možné si zvolit, jaký šablonovací systém má webová aplikace používat. Na výběr jsou tři. Prvním je Chameleon, druhým poté Jinja2, kterou využívá i Flask, a třetím je Mako. Všechny tři šablonovací systémy poskytují základní ochranu proti XSS. Další vestavěnou bezpečnostní ochranou, kterou framework Pyramid nabízí, je obrana proti clickjackingu. Funguje zde podobným způsobem jako v případě Flasku. Stačí nastavit hlavičku 'X-Frame-Options' na požadovanou hodnotu. Pyramid má stejně jako Django, Flask a Tornado vestavěnou ochranu proti CSRF. [50]

8.2.6. Bottle

Bottle stejně jako CherryPy nabízí velice malou sadu bezpečnostních ochran proti běžným útokům. Bottle využívá svůj vlastní šablonovací systém, který poskytuje ochranu proti XSS. Kromě toho podporuje i jiné, jako je např. Mako, Jinja2, či Cheetah. Bottle bohužel nenabízí zabezpečení proti CSRF, DNS rebinding a ani proti dalším nejmenovaným typům útoků. Vše je zde zapotřebí implementovat vlastními silami. [44] Z tohoto důvodu komunita kolem frameworku vytvořila speciální balíček s názvem Bottle Utils, který obsahuje ochranu proti CSRF útokům, a kromě toho i další užitečné třídy a funkce. [51]

8.3. Pravidelnost aktualizací

Tato podkapitola se bude zabývat tím, jak pravidelně jsou jednotlivé frameworky aktualizovány. Intenzita aktualizací může být indikátorem aktivity vývoje, přidávání nových funkcí a provádění oprav.

8.3.1. Django

Django své major verze vydává přibližně každý půl rok až rok. Opravné aktualizace vychází každý měsíc. Poslední vydanou major verzí Djanga je verze 5.0, která byla vydána 4. prosince roku 2023. Tato aktualizace přidala podporu nejnovější verze Pythonu 3.12. S touto verzí také skončila podpora pro Python ve verzi 3.8 a 3.9. [52]

8.3.2. Flask

Hlavní verze frameworku Flask vychází přibližně každých 5 až 10 měsíců. Opravné aktualizace jsou poté vydávány v rozmezí několika dní až měsíců. Poslední major verzí Flasku je verze 3.0.0, která vyšla 30. září 2023. Podpora pro aktuální verzi Python 3.12 byla přidaná již v předchozí verzi 2.3.3 ze srpna 2023. [53]

8.3.3. Tornado

Vývojářský tým stojící za frameworkem Tornado vydává major verze svého frameworku přibližně každých 7 až 9 měsíců. Opravné verze vychází velice zřídka a většinou řeší nějaký zásadní bezpečnostní problém. Poslední major verze 6.4.0 byla vydána 28. listopadu 2023 a přinesla podporu nejnovější verze Pythonu 3.12. a zároveň je i poslední vydanou aktualizací v době psaní této práce. [42]

8.3.4. CherryPy

Major verze frameworku CherryPy vychází velmi nepravidelně. Někdy je nová verze vydána v řádech měsíců a jindy to trvá několik let. Poslední major verzí je verze 18.9.0, která byla vydána 13. prosince 2023. [54]

8.3.5. Pyramid

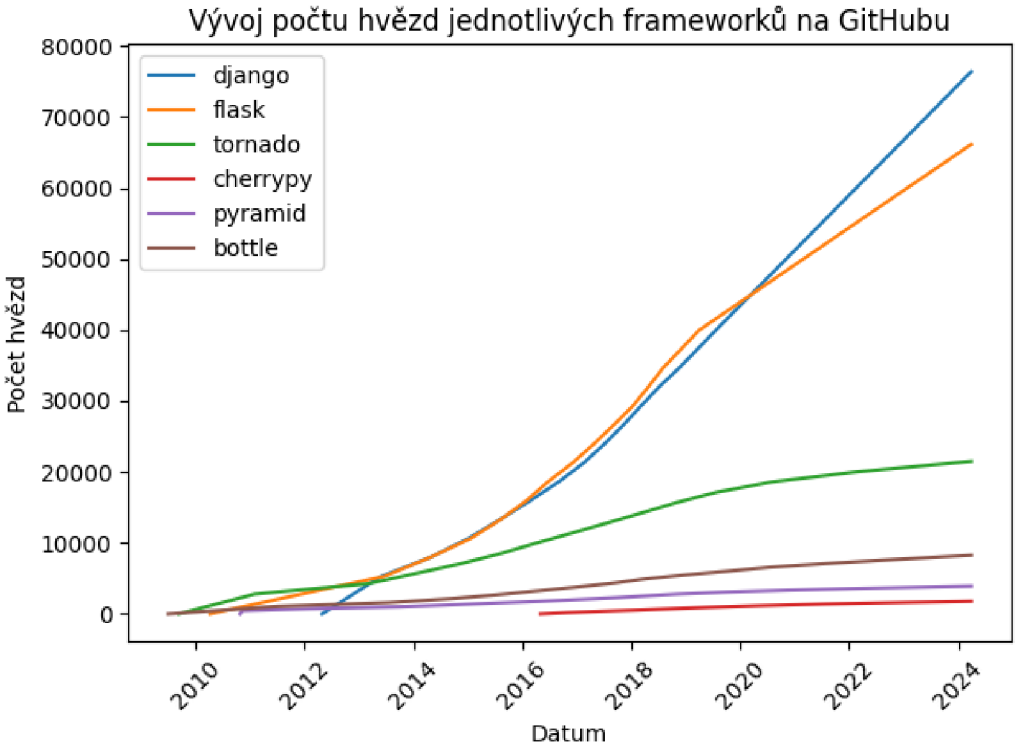
V posledních letech jsou nové major verze frameworku Pyramid vydávány v řádech let. Obvykle se jedná o opravdu rozsáhlé aktualizace, které přináší spoustu nových funkcí, odstraňují zastaralé funkce a nahrazují je funkcemi novými. Poslední vydanou major verzí je verze 2.0, která byla vydána 28. února 2021. Nejnovější opravná aktualizace 2.0.2 byla vydána v srpnu roku 2023. [55]

8.3.6. Bottle

Jednotlivé major verze pro framework Bottle obvykle vychází několik let od sebe. Poslední vydanou major verzí je verze 0.12.0, která byla vydána 3. prosince 2013. Opravné aktualizace vychází pravidelně v rozmezí několika měsíců až jednoho roku. [56] V současné době je ve vývoji verze 0.13, která zatím nemá datum vydání a jejím hlavním cílem je přidání podpory nových verzí Pythonu. Zároveň dojde k ukončení podpory verzí starších než Python 3.6. [57]

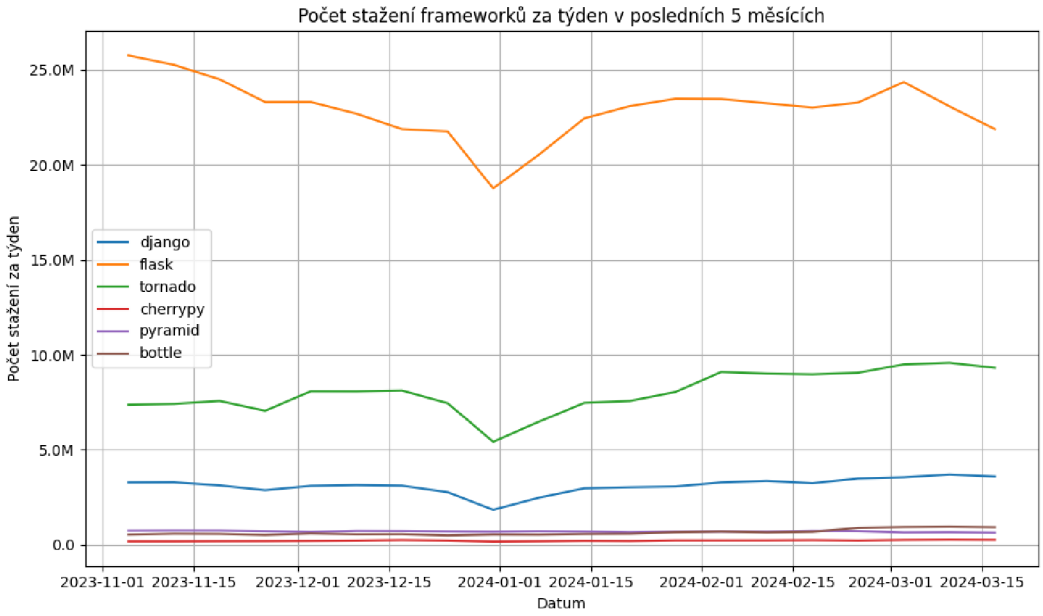
8.4. Sociální metriky

Graf č. 1 vyjadřuje vývoj počtu hvězd jednotlivých frameworků na platformě GitHub. V současné době je nejoblíbenějším webovým frameworkem pro Python Django s počtem přibližně 76 500. Druhým v pořadí je Flask s počtem hvězd 66 200. V období od roku 2016 do roku 2020 byl tento framework dokonce na prvním místě v žebříčku oblíbenosti. V současné době však jeho počet hvězd neroste takovým tempem jako tomu je u frameworku Django. Třetím nejoblíbenějším je framework Tornado s počtem 21 500 hvězd. Zbylé tři frameworky Bottle, Pyramid a CherryPy nedosahují ani 10 000 hvězd. Tempo růstu u těchto frameworků též není tak výrazné jako např. u Djanga či Flasku a z dlouhodobého pohledu se tak jedná o méně oblíbené projekty.



Graf 1. Vývoj počtu hvězd frameworků na GitHubu, Zdroj: vlastní

Graf č. 2 znázorňuje průměrný počet stažení frameworků od října 2023 do března 2024. Data v grafu jsou zprůměrovaná v týdenních intervalech. Nejstahovanějším frameworkem v tomto období byl Flask s průměrným počtem 22 719 519 stažení týdně. Byl téměř třikrát stahovanější než druhé Tornado. Nejméně stahovaným byl poté CherryPy s průměrným počtem 212 710 stažení za týden. Ani u jednoho z frameworků nedocházelo ve sledovaném období k výrazným nárůstům či poklesům počtu stažení, vyjma období konce prosince 2023 a začátku ledna 2024, které mohlo být způsobeno vánočními svátky.



Graf 2. Průměrný počet stažení frameworků za týden, Zdroj: vlastní

8.5. Výkon frameworků

Následující podkapitola se věnuje testování výkonu jednotlivých frameworků. Budou provedeny dva typy testů. Jedním z nich je load test, který ověřuje výkon aplikace pro specifikovaný počet uživatelů (200) v delším časovém intervalu (např. 15 minut). Druhým prováděným testem je spike test, který testuje výkon aplikace v krátkém intervalu (např. 3 minuty) pro určený počet uživatelů (např. 2 000).

8.5.1. Load Test

V této podkapitole jsou provedeny tři load testy pro každý z frameworků. Počet uživatelů jednotlivých load testů je 1, 20 a 200.

Způsob průběhu load testu je následující. Délka trvání testu je patnáct minut. V prvních dvou a půl minutách se zvýší počet virtuálních uživatelů z nuly na určený počet. Následujících deset minut se počet uživatelů drží na zvoleném počtu. Ve zbývajících dvou a půl minutách se počet uživatelů postupně sníží na nulu.

8.5.1.1. Pro 1 virtuálního uživatele

Výsledky load testu pro jednoho uživatele znázorněné v tabulce č. 1 ukazují, že jednotlivé frameworky dokázaly zpracovat 100 % všech požadavků, jejich celkový počet pro každý framework se pohyboval okolo 890. Průměrná doba vyřízení požadavku naznačuje, že nejrychleji dokázal zpracovat dotaz framework Tornado s hodnotou 1,18 ms. Naopak nejpomalejší byl framework Pyramid s časem zpracování požadavku 5,57 ms.

	Django	**Flask**	**Tornado**	**CherryPy**	**Pyramid**	**Bottle**
http_reqs	893	892	892	892	887	889
http_req_failed [%]	0	0	0	0	0	0
http_req_duration	1.81ms	2.44ms	1.18ms	2.17ms	5.57ms	4.71ms

Tabulka 1. Load test pro 1 virtuálního uživatele, Zdroj: vlastní

8.5.1.2. Pro 20 virtuálních uživatelů

Jak je vidět v tabulce č. 2, load test pro dvacet uživatelů odhalil, že všechny frameworky s výjimkou Pyramidu úspěšně zvládly zpracovat všechny požadavky. U frameworku Pyramid došlo k selhání v případě 0,01 % požadavků. Celkový počet požadavků pro jednotlivý framework se pohyboval v průměru okolo 14909 požadavků. V tomto případě dokázal nejrychleji zpracovat dotaz framework Pyramid a to rychlostí 1,81 ms. Daleko od této hodnoty nebyl ani Bottle se svou rychlostí 2.1 ms. Výrazně pomaleji potom zpracovával požadavky framework Django, který byl nejpomalejší a rychlost vyřízení požadavku dosahovala 11,88 ms.

	Django	**Flask**	**Tornado**	**CherryPy**	**Pyramid**	**Bottle**
http_reqs	14816	14919	14917	14917	14945	14945
http_req_failed [%]	0	0	0	0	0.01	0
http_req_duration	11.88ms	3.54ms	3.11ms	3.84ms	1.81ms	2.1ms

Tabulka 2. Load test pro 20 virtuálních uživatelů, Zdroj: vlastní

8.5.1.3. Pro 200 virtuálních uživatelů

Výsledky testu pro dvě stě virtuálních uživatelů, které jsou znázorněné v tabulce č. 3 ukazují, že v tomto případě pouze tři z šesti frameworků dokázaly zpracovat všechny požadavky. Frameworky CherryPy, Pyramid a Bottle měly neúspěšnost zpracování požadavků větší než čtyřicet procent. Počet požadavků pro každý framework se pohyboval v průměru okolo 148 000. Z důvodu velké neúspěšnosti požadavků u třech z frameworků je nelze brát jako kvalitní zdroj rychlosti zpracování požadavku, proto bude rychlost porovnávána pouze u těch s vysokou úspěšností. Z těchto tří úspěšných si nejlépe vedl Tornado, který dokázal zpracovat požadavek průměrně za 5,75 ms. Dalším v pořadí byl Flask s rychlostí 9,76 ms a nejpomalejším poté byl Django s rychlostí vyřízení požadavku za 28,73 ms.

	Django	**Flask**	**Tornado**	**CherryPy**	**Pyramid**	**Bottle**
http_reqs	145423	148238	148538	148643	148753	148671
http_req_failed [%]	0	0	0	41.5	47.84	43.27
http_req_duration	28.73ms	9.76ms	5.75ms	6.15ms	2.86ms	2.98ms

Tabulka 3. Load test pro 200 virtuálních uživatelů, Zdroj: vlastní

Výsledky jednotlivých load testů provedených v této podkapitole ukazují, že frameworky Django, Flask a Tornado nezaznamenaly problém v žádném z testovaných scénářů. Díky tomu lze říci, že by mohly být vhodnými kandidáty pro aplikace vyžadující stabilitu při větším množství současně připojených uživatelů.

8.5.2. Spike Test

V této podkapitole jsou provedeny dva spike testy pro každý z frameworků. Počet virtuálních uživatelů pro jednotlivé spike testy je 1000 a 2000.

Spike test spočívá v krátkodobém zvýšení počtu uživatelů, kteří zároveň vytváří požadavky na aplikaci, za účelem otestování, jak se s touto zvýšenou zátěží systém vypořádá. Délka testu je tři minuty. V první půl minutě se počet virtuálních uživatelů postupně zvyšuje z nuly na požadovaný počet a poté se tento počet udržuje po dobu dvou minut. Dále následuje půlminutový interval, ve kterém se počet uživatelů opět postupně snižuje na nulu.

8.5.2.1. Pro 1000 virtuálních uživatelů

Testovací výsledky pro tisíc virtuálních uživatelů, které jsou prezentovány v tabulce č. 4, demonstrují, že ani jeden z testovaných frameworků nebyl schopen efektivně zvládnout všechny přijaté požadavky. Nejlépe si vedly Django s Flaskem. Jejich neúspěšnost byla menší než jedno procento. Zbytek frameworků měl neúspěšnost vyšší než šedesát procent. Z toho Tornado dokonce více než devadesát procent. To je však způsobeno tím, že framework není plně podporován na operačním systému Microsoft Windows. Při větším zatížení začne Tornado vykazovat problémy s deskriptory souborů. Na webové stránce GitHub se k tomuto problému vyjádřil jeden z předních vývojářů. Uvedl, že z tohoto důvodu není framework plně podporován na operačním systému Windows. Dále konstatoval, že Windows nenabízí jinou alternativu, kterou by vývojáři mohli využít. Vývojář doporučuje využití jiných operačních systémů pro tento framework. [58] Zde lze porovnat pouze Django a Flask. Flask si v testu rychlosti zpracování požadavku vede skoro čtyřikrát lépe než Django, ale nedosahuje takové úspěšnosti ve zpracování požadavků.

	Django	**Flask**	**Tornado**	**CherryPy**	**Pyramid**	**Bottle**
http_reqs	106592	136151	111422	149013	149319	149321
http_req_failed [%]	0.08	0.77	96.3	62.79	68.11	66.19
http_req_duration	411.54ms	104.74ms	343.22ms	3.44ms	1.62ms	1.43ms

Tabulka 4. Spike test pro 1000 virtuálních uživatelů, Zdroj: vlastní

8.5.2.2. Pro 2000 virtuálních uživatelů

Výsledky testu pro dva tisíce virtuálních uživatelů znázorněné v tabulce č. 4 ukazují, že všechny frameworky kromě Djanga mají neúspěšnost zpracování požadavků okolo 60 % nebo vyšší. Z toho důvodu nemusí jejich hodnoty rychlosti zpracování úplně odpovídat realitě. Jediné Django se drží pod 1 % neúspěšnosti, a to ho v tomto testu i přes jeho pomalejší rychlost zpracování požadavku 912,93 ms činí jednoznačně nejspolehlivějším frameworkem.

	Django	**Flask**	**Tornado**	**CherryPy**	**Pyramid**	**Bottle**
http_reqs	157515	283026	260471	298207	298579	298573
http_req_failed [%]	0.33	59.51	99.15	70.45	72.77	72.37
http_req_duration	912.93ms	61.29ms	146.77	2.68ms	1.11ms	1.11ms

Tabulka 5. Spike test pro 2000 virtuálních uživatelů, Zdroj: vlastní

Výsledky jednotlivých spike testů provedených v této podkapitole ukazují, že nejspolehlivějším frameworkem z hlediska neúspěšnosti zpracování požadavků se jeví Django. To ani v jednom z testů nepřekročilo hranici jednoho procenta. Druhým nejlepším je pak Flask, který v prvním testu s tisíci uživateli poměrně obstál, ale ve druhém se dvěma tisíci uživateli si vedl již poněkud hůře. Vzhledem k velké neúspěšnosti zpracování požadavků u ostatních frameworků, tedy nelze spolehlivě porovnat jejich rychlost.

9. Výsledky porovnání

Tato kapitola se zabývá celkovým zhodnocením všech frameworků na základě použitých metodik z předchozí kapitoly.

9.1. Django

Dokumentace frameworku Django je kvalitně a přehledně popsaná. Obsahuje návody i pro začínající vývojáře a umožňuje přepínání mezi verzemi a jazyky. Dokumentaci je též možné stáhnout do offline režimu. Pokud jde o bezpečnost, Django je odolné vůči nejběžnějším útokům. Pravidelné aktualizace funkcí jsou prováděny jednou za půl roku až rok. Opravné aktualizace vycházejí každý měsíc. Na GitHubu jde o nejoblíbenější webový framework v Pythonu. V počtu stažení mezi porovnávanými frameworky však zaujímá až třetí pozici. Výkonností testy pak ukázaly, že i přesto, že se nejedná o nejefektivnější framework z pohledu rychlosti, tak je velmi spolehlivý.

9.2. Flask

Flask má přehlednou a dobře sepsanou dokumentaci. Obsahuje návod pro vzorovou aplikaci a je přístupná i pro nové vývojáře. Flask bohužel neumožňuje stažení dokumentace pro offline použití. Framework má ochranu proti nejběžnějším útokům. Pravidelné aktualizace funkcí vycházejí v intervalu přibližně pěti až deseti měsíců. Opravné jsou poté vydávány v rámci dnů až měsíců. Jedná se o druhý nejoblíbenější webový framework v Pythonu. Ve vztahu k počtu stažení zaujímá první příčku. Flask si také vedl poměrně dobře ve výkonnostních testech. Byl rychlejší než Django, ale v případě většího množství uživatelů byl méně spolehlivý než Django.

9.3. Tornado

Dokumentace frameworku Tornado je stručně a srozumitelně popsána, ale může být méně přehledná z důvodu obtížnější navigace mezi jednotlivými stránkami. Dokumentace je dostupná pouze v anglickém jazyce, avšak lze si stáhnout její offline podobu. Tornado má zabudovanou ochranu před běžnými webovými útoky. Samotný framework je aktualizován pravidelně v rozmezí každých sedmi až devíti měsících. Opravné aktualizace

pro dané verze nejsou zaručené a jsou vydávány zřídka. Tornado si drží třetí místo v popularitě na GitHubu a je druhým nejstahovanějším hned po frameworku Flask. Z výkonnostních testů lze vidět, že samotný framework je vcelku svižný a v případě menší zátěže i stabilní. Testy také ukázaly, že Tornado není plně podporován na operačním systému Windows.

9.4. CherryPy

CherryPy obsahuje dokumentaci, která je z hlediska složitosti přístupná pro nováčky. Je velice přehledná, dobře strukturovaná a vhodná pro vývojáře s různou úrovní znalostní. Dokumentace je dostupná pouze pro nejnovější verzi a nenabízí offline variantu. Framework má jen malou sadu bezpečnostních funkcí, které nezahrnují ani ochranu před CSRF útoky. CherryPy nemá pravidelné aktualizace, jednotlivé verze jsou od sebe vzdálené v řádech měsíců, ale mnohdy i let. Framework se netěší velké oblibě na GitHubu a ani v počtu stažení. Z výkonnostních testů si lze všimnout, že CherryPy je v případě malé zátěže spolehlivý, ale při větší zátěži dosahuje poměrně velké neúspěšnosti.

9.5. Pyramid

Navigace v dokumentaci frameworku Pyramid není moc přehledná a jednotlivé kapitoly jsou stručně popsané s velkým množstvím málo popsaného kódu. Dokumentace je dostupná pouze v angličtině a je stažitelná i pro offline použití. Pyramid nabízí důležité bezpečnostní funkce. Pravidelné aktualizace jsou prováděny v delších časových intervalech, avšak jsou hodně obsáhlé. Na GitHubu je tento framework méně oblíbený a nevykazuje vysoká čísla týdenního počtu stažení v porovnání např. s Flaskem. Výkonnostní testy ukázaly, že i při poměrně nízké zátěži nezvládl zpracovat všechny dotazy.

9.6. Bottle

Dokumentace frameworku Bottle je poměrně stručná, ale obsahuje podrobný návod, který je dostačující pro pochopení všech důležitých aspektů tvorby webových aplikací. Framework nezahrnuje mnoho bezpečnostních funkcí a většinu ochrany je zapotřebí implementovat vlastními silami nebo využít externích knihoven. Opravné aktualizace pro Bottle vychází pravidelně v rozmezí měsíců až roku, avšak poslední významná aktualizace vyšla naposledy v roce 2013. Z hlediska počtu sledujících na Githubu a množství stažení patří Bottle mezi méně oblíbené frameworky. Ve výkonnostních testech si vedl srovnatelně s frameworkem Pyramid. Při nejnižší zátěži byl rychlý, ale při mírně vyšší zátěži vykazoval velkou nespolehlivost.

10. Závěr

Cílem práce bylo seznámení s frameworky pro tvorbu webových aplikací v Pythonu a jejich porovnání.

Bakalářská práce začala úvodem do programovacího jazyka Python. Dále navázala vysvětlením rozdílu mezi frameworky a knihovnami. Následující kapitola popsala webové aplikace. Zbylá část práce se pak věnovala představení frameworků, jejich porovnání a následnému zhodnocení výsledku porovnání.

Práce ukázala, že každý z testovaných frameworků má své výhody a nevýhody. Po jejich porovnání může vývojář zvolit ten, který bude jeho situaci vyhovovat nejvíce. Frameworky Django a Flask prokázaly, že jejich pověst jakožto spolehlivých a kvalitních nástrojů pro tvorbu webových aplikací je opodstatněná. Jejich popularita neustále roste a pro vývojáře jsou příjemným a spolehlivým nástrojem pro tvorbu webových aplikací. Ostatní frameworky si ve výsledku též nevedly špatně. Řeší problémy svým specifickým způsobem, což může být v určitých situacích prospěšné. V kontextu menších projektů nebo experimentů by mohly být tyto frameworky považovány za vhodné kandidáty.

11. Seznam literatury

[1] KAUSHIK, K. Internal working of Python. *medium.com* [online]. 21. červenec 2021 [vid. 2024-01-31]. Dostupné z: https://medium.com/@kaushik.k/internal-working-of-python-415572929e7a

[2] PYTHON SOFTWARE FOUNDATION. 6. Modules. *docs.python.org* [online]. 30. leden 2024 [vid. 2024-01-31]. Dostupné z: https://docs.python.org/3/tutorial/modules.html

[3] MARK PILGRIM. 6.4. Using sys.modules. *Dive Into Python* [online]. 2004 [vid. 2024-01-31]. Dostupné z: https://linux.die.net/diveintopython/html/file_handling/more_on_modules.html

[4] MATTHES, Eric. *Python crash course: a hands-on, project-based introduction to programming*. 2nd edition. San Francisco, CA: No Starch Press, 2019. ISBN 978-1-59327-928-8.

[5] AMAZON WEB SERVICES, INC. What is Python? *aws.amazon.com* [online]. 2024 [vid. 2024-01-31]. Dostupné z: https://aws.amazon.com/what-is/python/

[6] VIVEKKOTHARI. Internal working of Python. *geeksforgeeks.org* [online]. 10. červenec 2023 [vid. 2024-01-31]. Dostupné z: https://www.geeksforgeeks.org/internal-working-of-python/

[7] CORY STIEG. Celebrating 32 Years Of Python With All The Reasons Why It's So Iconic. *codecademy.com* [online]. 20. únor 2023 [vid. 2024-01-31]. Dostupné z: https://www.codecademy.com/resources/blog/history-of-python-coding-language/

[8] JAVLER CANALES LUNA. Python 2 vs 3: Everything You Need to Know. *datacamp.com* [online]. červenec 2023 [vid. 2024-01-31]. Dostupné z: https://www.datacamp.com/blog/python-2-vs-3-everything-you-need-to-know

[9] What is Python Used For? 7 Real-Life Python Uses. *datacamp.com* [online]. listopad 2022 [vid. 2024-01-31]. Dostupné z: https://www.datacamp.com/blog/what-is-python-used-for

[10] AMAZON WEB SERVICES, INC. What is a Framework in Programming and Engineering? *aws.amazon.com* [online]. 2024 [vid. 2024-01-31]. Dostupné z: https://aws.amazon.com/what-is/framework/

[11] AMAZON WEB SERVICES, INC. What Is A Web Application? *AWS* [online]. 2023 [vid. 2023-06-30]. Dostupné z: https://aws.amazon.com/what-is/web-application/

[12] CODECADEMY TEAM. What is a Web App? *Codeacademy* [online]. 2023 [vid. 2023-06-30]. Dostupné z: https://www.codecademy.com/article/what-is-a-web-app

[13] MAMGAIN, Dinesh. *Single-page application vs Multi-page application: What Is Better for Your Project?* [online]. 29. listopad 2022 [vid. 2023-06-30]. Dostupné z: https://www.linkedin.com/pulse/single-page-application-vs-multi-page-what-better-your-

[14] CHADDHA, Manvi. Single Page Apps vs. Multi-Page Apps. *Coding Ninjas* [online]. 30. červen 2023 [vid. 2023-06-30]. Dostupné z: https://www.codingninjas.com/codestudio/library/single-page-apps-vs-multi-page-apps

[15] SKÓLSKI, Paweł. Single-Page Application vs. Multiple-Page Application. *Neoteric* [online]. 12. leden 2016 [vid. 2023-06-30]. Dostupné z: https://neoteric.eu/blog/single-page-application-vs-multiple-page-application

[16] KIRSTENS. Cross Site Scripting (XSS). *OWASP Foundation* [online]. 22. listopad 2022 [vid. 2023-06-30]. Dostupné z: https://owasp.org/www-community/attacks/xss/

[17] MOZILLA CORPORATION. HTML: HyperText Markup Language. *MDN Web Docs* [online]. 15. duben 2023 [vid. 2023-06-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ul

[18] MOZILLA CORPORATION. Doctype. *MDN Web Docs* [online]. 8. červen 2023 [vid. 2023-06-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/Doctype

[19] MICHÁLEK, Martin. Vše o meta značce pro viewport. *Vzhurudolu* [online]. 1. březen 2023 [vid. 2023-06-30]. Dostupné z: https://www.vzhurudolu.cz/prirucka/viewport-meta

[20] MOZILLA CORPORATION. CSS basics. *MDN Web Docs* [online]. 24. únor 2023 [vid. 2023-06-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics

[21] MOZILLA CORPORATION. JavaScript basics. *MDN Web Docs* [online]. 17. duben 2023 [vid. 2023-06-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

[22] CODECADEMY TEAM. Back-End Web Architecture. *codecademy.com* [online]. 2024 [vid. 2024-01-31]. Dostupné z: https://www.codecademy.com/article/back-end-architecture

[23] AMAZON WEB SERVICES, INC. What is Middleware? *aws.amazon.com* [online]. 2024 [vid. 2024-01-31]. Dostupné z: https://aws.amazon.com/what-is/middleware/

[24] ORACLE. What Is a Database? *oracle.com* [online]. 2023 [vid. 2024-01-31]. Dostupné z: https://www.oracle.com/database/what-is-database/

[25] SHAW, Ben, Saurabh BADHWAR, Andrew BIRD, BHARATH CHANDRA K S a Chris GUEST. *Web development with Django: learn to build modern web applications with a Python-based framework*. Birmingham: Packt Publishing, 2021. ISBN 978-1-83921-250-5.

[26] MATT MAKAI. Object-relational Mappers (ORMs). *Full Stack Python* [online]. 2022 [vid. 2024-01-31]. Dostupné z: https://www.fullstackpython.com/object-relational-mappers-orms.html

[27] RITVIK GUPTA a ANKIT SAHU. Software Architecture Patterns: What Are the Types and Which Is the Best One for Your Project. *turing.com* [online]. 15. říjen 2023 [vid. 2024-01-31]. Dostupné z: https://www.turing.com/blog/software-architecture-patterns-types/#Software_Architecture_Pattern

[28] TEJASWI CHAUDHARI. MVC vs MVT Architectural Pattern. *medium.com* [online]. 28. září 2021 [vid. 2024-01-31]. Dostupné z: https://medium.com/dsc-umit/mvc-vs-mvt-architectural-pattern-d306a56dce55

[29] MOZILLA CORPORATION. Django introduction. *Resources for Developers, by Developers* [online]. 2024 [vid. 2024-01-31]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

[30] RUBIO, Daniel. *Beginning Django: Web Application Development and Deployment with Python*. New York: Apress, 2017. ISBN 978-1-4842-2786-2.

[31] GRINBERG, Miguel. *Flask web development: developing web applications with Python*. 2nd edition. Sebastopol, California: O'Reilly, 2018. ISBN 978-1-4919-9173-2.

[32] What is WSGI? *wsgi.readthedocs.io* [online]. [vid. 2024-01-31]. Dostupné z: https://wsgi.readthedocs.io/en/latest/what.html

[33] DYLAN CASTILLO. Develop Data Visualization Interfaces in Python With Dash. *realpython.com* [online]. 20. únor 2023 [vid. 2024-01-31]. Dostupné z: https://realpython.com/python-dash/

[34] THE TORNADO AUTHORS. Tornado Web Server. *tornadoweb.org* [online]. 2024 [vid. 2024-01-31]. Dostupné z: https://www.tornadoweb.org/en/stable/

[35] CHERRYPY TEAM. CherryPy — A Minimalist Python Web Framework. *docs.cherrypy.dev* [online]. 2024 [vid. 2024-01-31]. Dostupné z: https://docs.cherrypy.dev/en/latest/index.html

[36] HELLEGOUARCH, Sylvain. *CherryPy essentials: Rapid Python Web application development ; design, develop, test, and deploy your Python web applications*

easily. 1. publ. Birmingham: Packt Publ, 2007. From technologies to solutions. ISBN 978-1-904811-84-8.

[37] CHERRYPY TEAM. Why CherryPy? *docs.cherrypy.dev* [online]. 2024 [vid. 2024-01-31]. Dostupné z: https://docs.cherrypy.dev/en/latest/intro.html

[38] AGENDALESS CONSULTING. Pyramid Introduction. *docs.pylonsproject.org* [online]. 25. červenec 2023 [vid. 2024-01-31]. Dostupné z: https://docs.pylonsproject.org/projects/pyramid/en/2.0-branch/narr/introduction.html

[39] MARCEL HELLKAMP. Bottle: Python Web Framework. *bottlepy.org* [online]. 15. prosinec 2022 [vid. 2024-01-31]. Dostupné z: https://bottlepy.org/docs/dev/

[40] DJANGO SOFTWARE FOUNDATION. Django documentation. *Django documentation* [online]. 2024 [vid. 2024-04-07]. Dostupné z: https://docs.djangoproject.com/en/5.0/

[41] PALLETS. Flask documentation. *Flask* [online]. 2010 [vid. 2024-04-07]. Dostupné z: https://flask.palletsprojects.com/en/3.0.x/

[42] THE TORNADO AUTHORS. Tornado documentation. *Tornado Web Server* [online]. 2024 [vid. 2024-04-07]. Dostupné z: https://www.tornadoweb.org/en/stable/index.html#

[43] AGENDALESS CONSULTING. Pyramid documentation. *The Pyramid Web Framework* [online]. 2023 [vid. 2024-04-07]. Dostupné z: https://docs.pylonsproject.org/projects/pyramid/en/2.0-branch/

[44] MARCEL HELLKAMP. Bottle documentation. *Bottle: Python Web Framework* [online]. 15. prosinec 2022 [vid. 2024-04-07]. Dostupné z: https://bottlepy.org/docs/dev/index.html

[45] Security in Django | Django documentation. *Django Project* [online]. [vid. 2024-04-08]. Dostupné z: https://docs.djangoproject.com/en/5.0/topics/security/

[46] *Security Considerations — Flask Documentation (3.0.x)* [online]. [vid. 2024-04-12]. Dostupné z: https://flask.palletsprojects.com/en/3.0.x/web-security/

[47] *X-Frame-Options - HTTP | MDN* [online]. 12. prosinec 2023 [vid. 2024-04-12]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

[48] *Authentication and security — Tornado 6.4 documentation* [online]. [vid. 2024-04-12]. Dostupné z: https://www.tornadoweb.org/en/stable/guide/security.html

[49] *Advanced — CherryPy 18.9.1.dev30+g1073610.d20240225 documentation* [online]. [vid. 2024-04-12]. Dostupné z: https://docs.cherrypy.dev/en/latest/advanced.html

[50] *Security — The Pyramid Web Framework v2.0.2* [online]. [vid. 2024-04-12]. Dostupné z: https://docs.pylonsproject.org/projects/pyramid/en/latest/narr/security.html

[51] *Bottle Utils documentation — Bottle Utils 2.0.dev1 documentation* [online]. [vid. 2024-04-12]. Dostupné z: https://bottle-utils.readthedocs.io/en/latest/index.html

[52] Release notes | Django documentation. *Django Project* [online]. [vid. 2024-04-12]. Dostupné z: https://docs.djangoproject.com/en/5.0/releases/

[53] *Changes — Flask Documentation (3.0.x)* [online]. [vid. 2024-04-12]. Dostupné z: https://flask.palletsprojects.com/en/3.0.x/changes/

[54] *History — CherryPy 18.9.1.dev30+g1073610.d20240225 documentation* [online]. [vid. 2024-04-12]. Dostupné z: https://docs.cherrypy.dev/en/latest/history.html

[55] *The Pyramid Web Framework — The Pyramid Web Framework v2.0.2* [online]. [vid. 2024-04-12]. Dostupné z: https://docs.pylonsproject.org/projects/pyramid/en/2.0-branch/index.html#change-history

[56] Tags · bottlepy/bottle. *GitHub* [online]. [vid. 2024-04-12]. Dostupné z: https://github.com/bottlepy/bottle

[57] *Release Notes and Changelog — Bottle 0.13-dev documentation* [online]. [vid. 2024-04-12]. Dostupné z: https://bottlepy.org/docs/dev/changelog.html

[58] too many file descriptors in select() · Issue #1802 · tornadoweb/tornado. *GitHub* [online]. [vid. 2024-04-12]. Dostupné z: https://github.com/tornadoweb/tornado/issues/18

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**David Malík**
Studium:	I2100239
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Webové frameworky pro Python**
Název bakalářské práce AJ:	Python Web Frameworks

Cíl, metody, literatura, předpoklady:

Cíl: Představit a porovnat frameworky pro tvorbu webových aplikací v jazyce Python.

Zadávající pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	doc. Mgr. Tomáš Kozel, Ph.D.
Datum zadání závěrečné práce:	10.10.2023