VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA


# WEBOVÝ NÁSTROJ PRO POČÍTÁNÍ ROZSAHU TEXTU V NORMOSTRANÁCH

WEB TOOL FOR ESTIMATION OF DOCUMENT SIZE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS


AUTOR PRÁCE — SIMONA DLOUHÁ
AUTHOR

VEDOUCÍ PRÁCE — Prof. Ing. ADAM HEROUT, Ph.D.
SUPERVISOR



BRNO 2019

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

# Zadání bakalářské práce

21395

Studentka: **Dlouhá Simona**
Program: Informační technologie
Název: **Webový nástroj pro počítání rozsahu textu v normostranách**
**Web Tool for Estimation of Document Size**
Kategorie: Uživatelská rozhraní
Zadání:

1. Seznamte se s problematikou tvorby jednoduchých webových služeb.
2. Seznamte se s formátem PDF a s nástroji pro jeho strojové zpracování.
3. Vyvíjejte algoritmy pro zpracování PDF se zaměřením na zjištění různých charakteristik souvisejících s jeho rozsahem - počítání znaků, přehled kapitol a podkapitol, citace, statistiky ohledně obrázků, atp.
4. Navrhněte a implementujte jednoduchou webovou službu, která umožní nahrát PDF a poskytne podrobný a užitečný souhrn informací souvisejících s jeho rozsahem.
5. Testujte navrženou službu, vyhodnoťte její chování a užitečnost a iterativně ji vylepšujte.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek pro prezentování projektu.

Literatura:

- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN-13: 978-0321965516
- Susan M. Weinschenk: 100 věcí, které by měl každý designér vědět o lidech, Computer Press, Brno 2012
- Jan Řezáč: Web ostrý jako břitva, Baroque Partners, 2014

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2, značné rozpracování bodů 3, 4 a 5.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

## Abstrakt

Cílem této práce je navrhnout a vytvořit webový nástroj, který uživateli umožní získat rozsah textu v normostranách z PDF souboru. Práce popisuje navržené algoritmy pro získávání statistik ze souboru. Nástroj je implementován pomocí frameworku Django, uživatelské rozhraní je realizováno pomocí HTML, CSS a JavaScriptu. Výsledkem této práce je nástroj, který poskytuje statistiky o textu, obrázcích a kapitolách, přínosem je také přesnější počítání normostran u dokumentů obsahujících obrázky ve srovnání s dalšími aplikacemi.



## Abstract

Aim of this thesis is to design and create web tool, which will allow users to get a text range in standard pages from a PDF file. The thesis introduces algorithms designed for retrieving statistics from a file. The tool is implemented using Python and Django framework, user interface is realized by HTML, CSS and JavaScript. Result of this thesis is a tool that provides text statistics, image statistics and chapters overview. Benefit of this tool is also more precise page counting in documents with images in comparison with other tools.



## Klíčová slova

normostrany, PDF, Django, webová aplikace, uživatelské rozhraní

## Keywords

standard pages, PDF, Django, web application, user interface


## Citace

DLOUHÁ, Simona. *Webový nástroj pro počítání rozsahu textu v normostranách.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Prof. Ing. Adam Herout, Ph.D.

# Webový nástroj pro počítání rozsahu textu v normostranách

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana prof. Ing. Adama Herouta, Ph.D. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

. . . . . .
Simona Dlouhá
15. května 2019


## Poděkování

Chtěla bych poděkovat panu prof. Ing. Adamovi Heroutovi, Ph.D. za konzultace, cenné rady a navádění správným směrem při řešení této práce. Dále patří velké díky i panu Michalu Kuželovi za jeho rady a podporu.

# Obsah

# Kapitola 1

# Úvod

Kolik mám napsáno normostran? To je otázka, kterou si klade mnoho studentů vysokých škol při dokončování jejich diplomových prací. Kolik normostran má tento student? Zní jedno z kritérií při hodnocení diplomových prací. Uživatelé LaTeX při prozkoumání internetu nenaleznou uspokojivé řešení. Microsoft Word svým uživatelům zobrazuje počet znaků v dokumentu, ale neposkytuje jim údaje o počtu normostran. K tomu se ještě přidává fakt, že žádný nástroj nepočítá normostrany z obrázků.

Náplní této bakalářské práce je vytvoření webového nástroje pro analýzu souborů ve formátu PDF. Tvorba nástroje je podložena předchozím studiem problematiky vytváření webových služeb a studiem formátu PDF. Při tvorbě tohoto nástroje jsem se zaměřila na dvě cílové skupiny – studenty, kteří tvoří diplomovou práci a vedoucí nebo oponenty, kteří ji kontrolují.

Vytvořený nástroj poskytuje uživatelům možnost snadného výpočtu normostran. Mimo počet normostran poskytuje také statistické údaje o textu, které mohou být užitečné při psaní práce i při hodnocení. Dále zobrazuje přehled o jednotlivých kapitolách, jejich rozsahu a počtu normostran. K tomu přidává ještě výpočet normostran z obrázků a jejich tabulkový přehled. Nástroj byl vystavován pomocí cloudové platformy Heroku a je dostupný na adrese `http://standardpages.herokuapp.com/standardpages/`. Během vývoje byly prováděny průběžné testy na uživatelích a na základě jejich hodnocení byly přidány do prostředí nové funkcionality související s analýzou dokumentu a také upraveny některé prvky uživatelského rozhraní.

Přínosem tohoto nástroje je snadná analýza celého souboru PDF včetně obrázků a souvisejících statistik, jediná akce od uživatele je výběr souboru. Zvláště výpočet obrázků a statistiky z textu byly oceněny během testování na uživatelích. Nalezené existující nástroje uvedené v této práci neposkytují analýzu obrázků, některé také vyžadují kopírování obsahu práce do textového pole.

V kapitole 2 se rozebírá proces tvorby webových stránek a je zde popsána teorie pro návrh aplikace a technologie použité při implementaci. Kapitola 3 představuje základy formátu PDF. Právě ze souborů ve formátu PDF jsou extrahovány data a prováděny analýzy. Další kapitola 4 popisuje návrh aplikace. Jsou zde uvedena zkoumaná existující řešení, popis cílové skupiny aplikace, stanovené požadavky na aplikaci, způsob zpracování souboru na serveru a návrh uživatelského rozhraní. Kapitola 5 se zaměřuje na vlastní implementaci nástroje, popisuje použité algoritmy a metody pro extrahování textu, obrázků a kapitol. V poslední kapitole 6 je popsán průběh testování a vyhodnocení uživatelského testování. Je zde také uvedeno porovnání přesnosti s ostatními nástroji zaměřujícími se na počítání normostran.

# Kapitola 2

# Tvorba webových stránek

V této kapitole jsou popsány základy tvorby webových stránek. Nejdříve jsou uvedeny způsoby návrhu aplikace. Dále jsou vysvětleny zvolené technologie pro implementaci řešení a vystavování. Jsou zde také krátce shrnuty technologie pro tvorbu klientské i serverové části, protokol pomocí kterého klient a server komunikují a jeden z možných způsobů zveřejňování stránek.

## 2.1 Návrh webu

Jak přistupovat k návrhu webu je otázkou, na kterou existuje mnoho názorů. V následujících podkapitolách jsou rozebrány dva možné přístupy, které lze kombinovat i dohromady. První rozděluje návrh do čtyř fází, druhý se na návrh kouká z psychologického hlediska.

### Jak navrhovat web

V knize Jana Řezáče [11] je proces návrhu rozložen do čtyř fází:

**Objevování** Tato fáze zahrnuje průzkum ohledně aplikace, která se bude vyvíjet. Je důležité mít vizi o výsledném produktu, znát kdo je náš klient, kdo jsou naši spojenci a konkurenti, co vyvíjíme za produkt, jak ho budeme vyvíjet a proč ho vyvíjíme. Měli bychom pokládat otázky správným lidem.

Každý web by měl mít jedinečný prodejní argument – *Unique Selling Propositon (USP)*, který by měl web odlišit od ostatních a zákazníkům ukázat proč právě tento produkt je ten který potřebují.

Klient by měl mít svou značku – *brand* dávající jí smysl v oku jeho zákazníků. Pokud zákazník uvidí značku, měla by se mu v hlavě asociovat s jejími atributy a benefity. Protože je web jedno z míst interakce se zákaznickou značkou, marketingová podpora webu je potřebná. Bez této podpory se snižuje šance na úspěch webu.

**Uživatelský výzkum** V průběhu této fáze se provádí různé uživatelské výzkumy. Provádí se analýza současného webu, konkurenčních nebo podobných webů. Dále se ověřují hypotézy pomocí dotazníkových průzkumů, výzkumu v terénu, atd.

Jedním z doporučených postupů je provádět tzv. *hloubkový rozhovor* s lidmi, kteří jsou vhodní pro projekt. Na tento rozhovor je potřeba být schopný se vcítit do pocitů respondenta a umět mu naslouchat. Díky příběhům by mělo být jednodušší si představit uživatele používající výsledný web.

**Návrh webu** Zahrnuje stanovení směru kterým se bude web ubírat, struktury a obsahu webu, prototypování a grafický návrh webu.

Směr se získá interpretací ideí z předchozích fází do idei o webu.

Poté, co se vytvoří struktura a navrhne obsah webu začnou se vytvářet prototypy webu. Je považováno za chybu vytvářet web bez textů a obrázků s tím, že je později klient sám doplní, protože může dojít k tomu, že rozložení webu nebude v pořádku. Navíc by se mělo jednat o originální texty a fotky aby došlo k zachování brandu.

Prototypování má tři fáze: *skicování*, *wireframy* a *prototyp*. Při *skicování* se generují nápady tím, že si člověk vezme tužku a papír a kreslí svoje nápady, tak jak mu přijdou do mysli. *Wireframy* nebo-li drátěné modely mají za cíl ukázat klientovy rozvržení jednotlivých stránek. Jejich součástí by měl být i obsah stránky a obsaženým prvkům by měla být přiřazena priorita. *Prototyp* je chápán jako wireframe provázaný s jinými wireframy. Uživatel po proklikání prototypu získá jasnější představu o tom jaký bude mít výsledný web vzhled a chování.

Jan Řezáč radí grafický návrh webu nepodceňovat, jak z hlediska vzhledu, tak z hlediska funkčnosti. Měl by být zachován brand klienta a to, jestli je zachován správně, se nedá zjistit bez analýzy uživatelů webu nebo uživatelským testováním.

**Evaluace** Tato fáze se týká zpětné vazby, která má zlepšit kvalitu řešení a případně opravit chyby nebo přidat/odebrat vlastnosti.

Na získání zpětné vazby lze použít například uživatelské testování dle Kruga. Princip této metody spočívá v tom, že se určí testovací scénáře. Tyto scénáře zkouší 3-5 uživatelů na prototypu či hotovém webu za přítomnosti dalších dvou a více osob. Jedna zadává scénáře, druhá pozoruje a zapisuje.

### Jak udělat web atraktivní

Susan Weinschenk se ve své knize [9] zaměřuje na web z hlediska uživatele. Jako psycholožka zkoumá, jak uživatel vnímá určité věci na webu a z toho vyvozuje závěry, kterými by se měl webový designér při návrhu držet, aby měl uživatel co nejlepší User Experience (uživatelský zážitek).

Kniha obsahuje deset zajímavých kapitol, z nichž tyto vybrané jsou relevantní k tématu této práce:

**Jak lidé vidí.** Zrak je jeden z nejpoužívanějších smyslů. Mozek každého člověka ale vidí stejnou věc jinak. To co návrháři může přijít jasné, někteří lidé nepochopí stejně, ale správnou prezentací dané věci se to dá do jisté míry ovlivnit.

Člověk se soustředí na to co je v centru a neměl by se příliš rozptylovat periferním viděním. V periferii stránky by tedy neměli být blikající animace, ale elementy s výstižným obsahem.

S. Weinschenk doporučuje používat shlukování elementů – vzory, protože je lidské oko hledá. Mezery by měli být větší mezi elementy, co k sobě nepatří, a menší mezi těmi, co k sobě patří.

**Jak lidé čtou** Lidé vnímají slova napsané velkými písmeny jako rozkazy, proto se nedoporučuje je používat až na menu, titulky nebo důležité upozornění před akcí. Velikost písma by měl být dobře čitelný pro všechny generace uživatelů. Text by měl být strukturovaný, protože to uživateli ulehčuje orientaci v něm.

**Jak lidé přemýšlí** Lidé věci rádi kategorizují – pokud se na stránce nachází mnoho informací, které nejsou roztříděné, člověk je zbytečně zmatený a cítí se přetížený tím, že věci musí sám kategorizovat. Je dobré informace organizovat tak, aby byly lépe pochopitelné.

**Lidé dělají chyby** Zde Weinschenk doporučuje přemýšlet o tom co uživatel pravděpodobně udělá za chyby, případně doporučuje při testování prototypu tyto chyby vypozorovat. Pro chyby by měla být jasná chybová hláška vysvětlující proč nastala a co udělat, aby nenastala.

## 2.2 HTML, CSS, JavaScript

**HyperText Markaup Language** nebo-li HTML [8] je jazyk používaný pro vytvoření kostry webu. Pomocí něj lze na webu strukturovat a vizualizovat různý obsah od textu po videa.

Jak značí slovo **hypertext** v jeho názvu, lze se pomocí něho pohybovat po webu a to pomocí kliknutí na *hyperlink*. Hyper v tomto slově znamená nelineární, protože díky hyperlinkům se lze dostat kamkoliv bez jakéhokoli řádu.

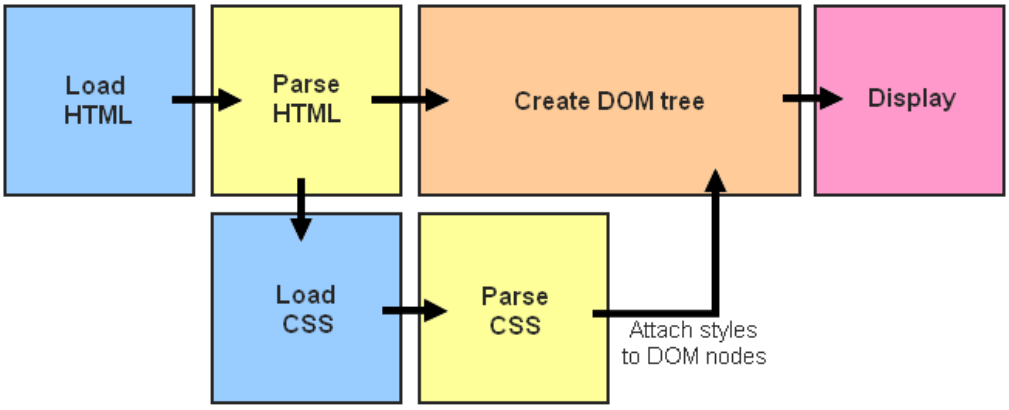
Slovo **markup** říká, že HTML je značkovací jazyk. Jeho syntax je tvořen pomocí *HTML tagů (značek)*. Webový prohlížeč pomocí těchto tagů po zpracování dokumentu HTML vytvoří strom DOM[1] elementů, které poté zobrazuje. Každý tag může mít hodnotu a atributy. Mezi významné atributy, které napomáhají při stylování webu patří *class* a *id*. Atribut *class* může mít i více hodnot, pomáhá označit elementy se stejnými vlastnostmi, stejnou hodnotu může tedy mít i více elementů v dokumentu. Atribut *id* je unikátním označením elementu.

**Cascading Style Sheets** [5] pomáhají strukturovanému HTML dodat styl a vytvořit tak prezentaci webu. Na obrázku 2.1 je zobrazen způsob, kterým prohlížeč připojuje styl k danému DOM elementu. Styl se přiřazuje jednotlivým elementům buďto v úseku CSS kódu nebo přímo v atributu elementu style. Pomocí selektorů lze vybrat HTML elementy, které mají mít daný styl.

**JavaScript** [6] je skriptovací jazyk, který se podílí na vytváření a ovládání dynamického obsahu webových stránek. Časté využívání velmi podobných funkcionalit v JavaScriptu na mnoho webech dalo vzniknout knihovně jQuery, která je založená právě na JavaScriptu. Zjednodušuje práci například s výběrem elementu nebo s přiřazením funkce, která má nastat po kliknutí na daném elementu.

Z hlediska webových stránek HTML představuje strukturu stránky. CSS vytváří této struktuře prezentaci a JavaScript prezentaci přidává chování. Společně tvoří základní technologie pro vytvoření stylované dynamické webové stránky.

[1] Document Object Model




Obrázek 2.1: Způsob jakým prohlížeč zobrazuje dokument HTML stylovaný pomocí CSS. Nejdříve je načten dokument HTML. Po jeho zpracovaní se načte CSS které je odkazované nebo obsažené v tomto HTML dokumentu a zpracuje se. Styly se připojí k DOM uzlům získaných z HTML. Tento strom DOM elementů je následně prohlížečem zobrazen.[3]

## 2.3 HTTP

**HyperText Transfer Protocol** [7] je protokol na aplikační vrstvě, který je používán pro přenos informací na webu. HTTP funguje na modelu *klient-server*, kde klient je webový prohlížeč a server je HTTP server. Prohlížeč posílá požadavky HTTP serveru a server zasílá prohlížeči odpovědi na tyto požadavky. Struktura požadavku i odpovědi se typicky skládá z hlaviček a obsahu (těla). Hlavičky jsou odděleny novým řádkem. V obsahu mohou být zaslány různé data, typ dat je určen hlavičkou Content-type. Server v odpovědi zahrnuje Response-code, díky kterému klient ví, zdali-se byl požadavek úspěsný (např. kód 200 – OK) nebo jestli při zpracování došlo k chybě (např. kód 404 – NOT FOUND).

Mezi základní *HTTP metody* patří GET, POST, PUT a DELETE. GET metoda se typicky používá při získávání dokumentu. Lze ji použít i na odeslání formuláře, kde se odeslaná data předávají pomocí proměnných v URL adrese. Kvůli předávání informací v URL adrese je metoda GET nevhodná pro odesílání citlivých dat. POST metoda se typicky používá na odeslání dat formuláře. Zde se data odesílají v těle požadavku. Pomocí metody POST lze zaslat větší množství informací a dokonce i soubory, což se u metody GET nedoporučuje.

## 2.4 Django

Django [1] je open source framework pro vývoj webových aplikací. Je napsaný v jazyce Python. Nabízí sadu komponent, které jsou užitečné při tvorbě webových stránek: řízení uživatelské autentizace, formuláře, způsob jak nahrávat soubory atd.

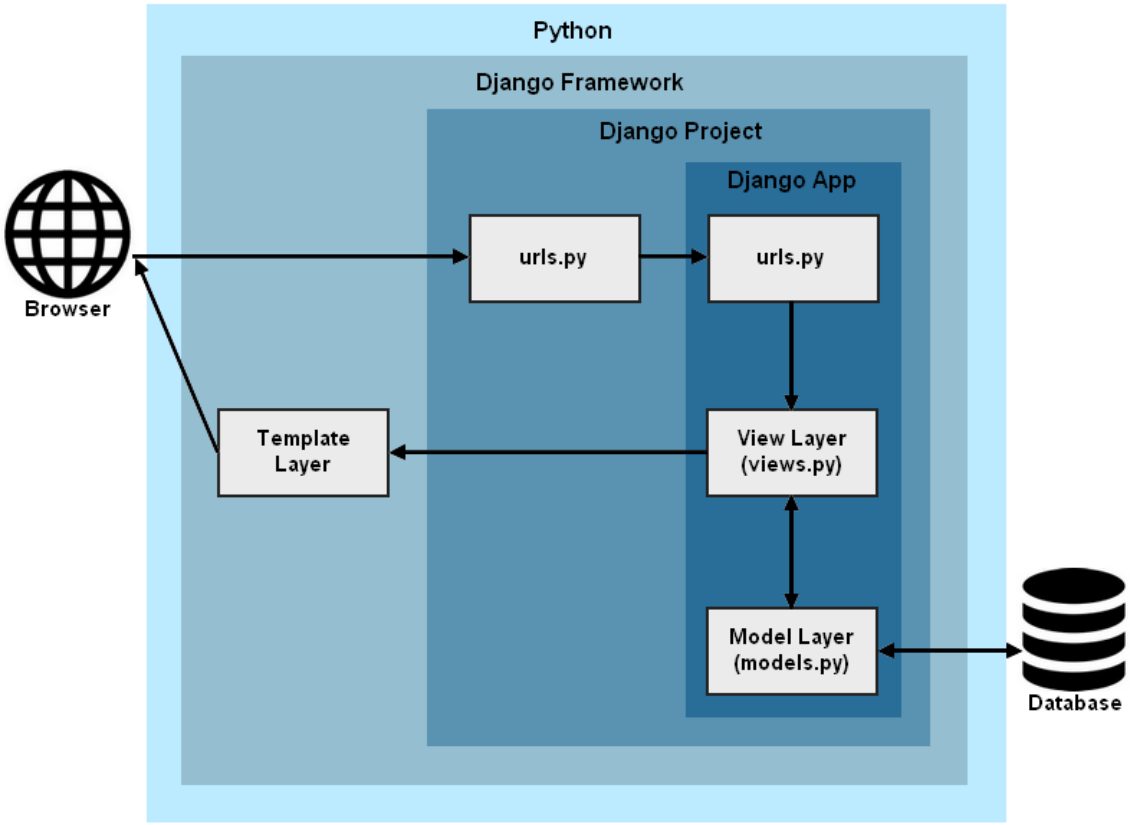
Jak je naznačeno na obrázku 2.2, framework Django je založený na architektuře **model-view-controller**[4]. Poté, co na server přijde požadavek od klienta, je předán frameworku Django [2], který pomocí **urlresolver** zjišťuje, jakou akci má vykonat a na kterou adresu

[3] Inspirováno: https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/How_CSS_works

[4] http://www.jdl.co.uk/briefings/MVC.pdf

požadavek předat. Prochází se seznam definovaných URL[5] v *urls.py* od shora dolů a pokud se některý záznam shoduje se záznamem z požadavku, framework Django předá požadavek do *view.py*. Ve view je možné přistupovat k datům z databáze přes datové modely definované v *models.py*. Proměnné z view se předají do HTML šablony, která se vyhodnotí a výsledný dokument se pošle jako odpověď klientu.

Django poskytuje možnost přednastaveného přístupu k databázi, kterou je možné změnit. Při vývoji také vytváří migrační skripty, které jsou nezávislé na technologii použité pro databázi.




Obrázek 2.2: Zjednodušený princip Django Frameworku: Webový prohlížeč (klient) pošle HTTP požadavek na server. URL požadavku je vyhodnocena pomocí urlresolver a předáno dál aplikaci. Zde se skrze view, které obousměrně komunikuje s modely propojené s databází, předají proměnné do HTML šablon, ze kterých vznikne výsledný dynamicky vygenerovaný HTML soubor, který se pošle jako odpověď na požadavek od prohlížeče.[7]

V šablonách HTML je použit speciální **Django template language**. Šablony obsahují proměnné a tagy:

- Do **proměnných** se dosadí hodnota při generování HTML souboru. Proměnné mají následující syntax, kde *variable* je název proměnné: `{{ variable }}`. Jméno proměnné se může skládat z alfanumerických znaků a podtržítka. Tečka ve jménu proměnné znamená že přistupujeme k atributu proměnné, tedy objektu.

[5] Uniform Resource Locator
[7] Inspirováno: https://medium.com/@cvarelaruiz/why-i-love-python-and-django-26596ce4d82e

- **Tagy** kontrolují logiku šablony. Jejich syntax je: `{% tag %}`, kde *tag* je obsah tagu. Některé tagy mají i ukončovací tagy. Kontrolují průběh vyhodnocování pomocí if, else, elif a smyček. V podmínkách nebo smyčkách se používají proměnné, lze tak například ověřit jestli je proměnná definovaná nebo iterovat skrze objekt.

## 2.5 Použité knihovny

Pro vykreslení grafů z polí hodnot slouží Python **knihovna matplotlib** [3]. Tato knihovna je napsaná v jazyce Python a je založená na emulaci MATLAB™. Umožňuje snadné vytvoření grafů, histogramů, korelačních diagramů které vypadají reprezentativně a lze je exportovat do různých formátů (například formát PNG).

Python **knihovna re**[8]je knihovna pracující s regulárními výrazy. Nabízí funkce pro vyhledání nebo nahrazení shody v řetězci pomocí daného regulárního výrazu. Regulární výraz stanovuje množinu řetězců, která mu odpovídá.

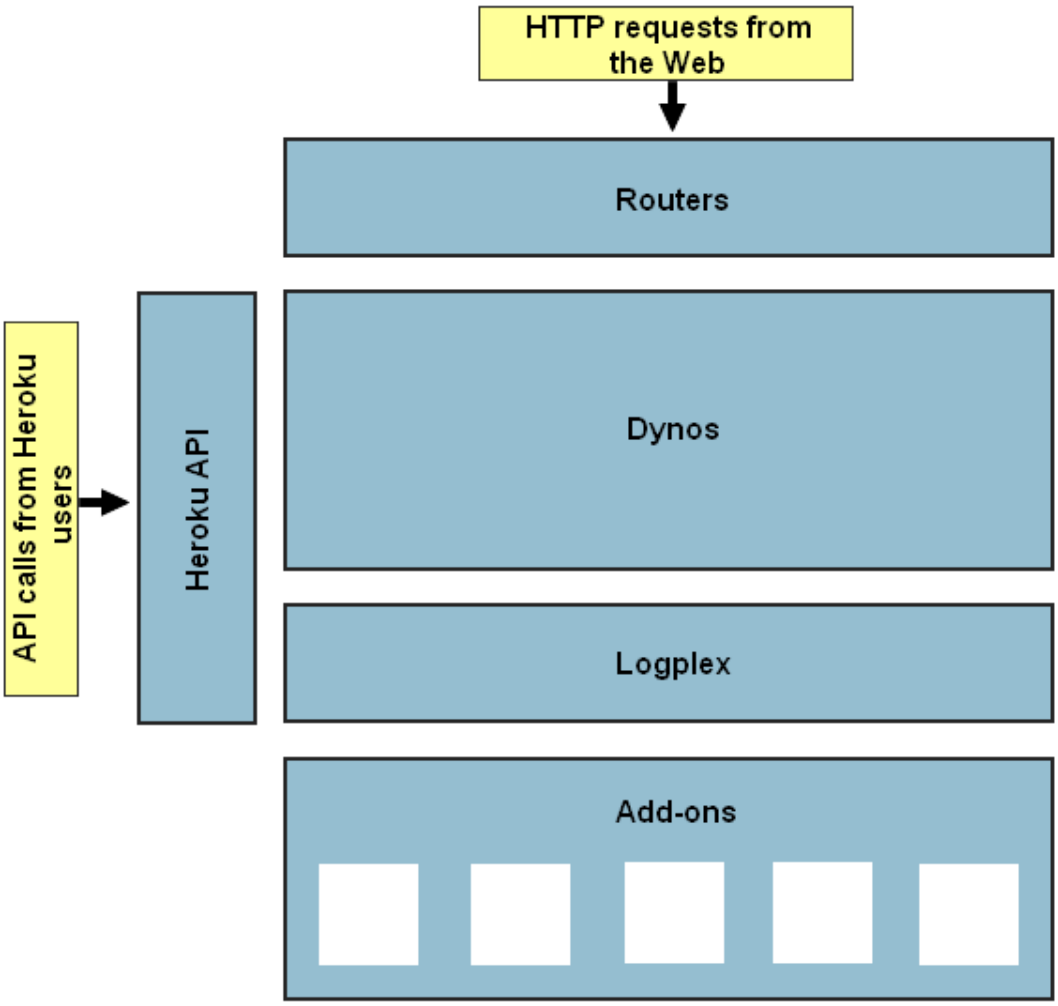
**Knihovna PyMuPDF**[9]je Python knihovna, která pracuje se soubory PDF. Umí zapsat obsah tohoto souboru do různých formátů jako HTML, XML nebo do Python slovníku. Nabízí metody pro získávání textu, obrázků a stránek souboru PDF. Mimo získávání obsahu již existujícího PDF, umí do souboru i zapisovat.

## 2.6 Heroku

Heroku [4] je prostředí pro nasazení a běh aplikací v cloudu – **Platform as service** (PaaS). Jedná se o cloudovou službu, která zastřešuje celou řadu dalších služeb, například propůjčuje výpočetní kapacitu. Podporuje mnoho programovacích jazyků, mimo jiné i Python. Umožňuje nahrávání aplikace na server s využitím verzovacích nástrojů jako Git[10], automatické sestavení aplikace na serveru a monitorovaní aplikace pomocí logů. Architekturu Heroku lze vidět na obrázku 2.3.

Po vystavení aplikace na server, běží aplikace v kontejneru zvaném **dyno**. Počet instancí aplikace běžících na serveru se odvíjí od počtu dyno. Dynos lze nastavit v souboru *Procfile* v kořenovém adresáři aplikace, lze zde nastavit například Python HTTP servery Gunicorn[11] nebo Waitress[12]. Dyno je jako malý virtuální počítač, ale protože může být rozdělen na několik reálných počítačů, jsou požadavky z webového rozhraní směřovány pomocí **routerů** a API[13] volání jsou zpracovány pomocí **Heroku API**. Router se chová jako brána mezi uživatelem a dynem ve kterém se vykonává kód aplikace. Výstupy serveru jsou zaznamenávány do logů pomocí **Logplexu**. Lze je prohlížet ve webovém rozhraní Heroku nebo je zobrazit s využitím terminálu. Mezi významné přídavné funkcionality patří zejména databáze. Tyto funkcionality nejsou přímou součástí Heroku, ale lze je získat instalací **Add-on programů**, které jsou přístupné přes add-ons API.

[8] https://docs.python.org/3/library/re.html
[9] https://github.com/pymupdf/PyMuPDF
[10] https://git-scm.com/
[11] https://gunicorn.org/
[12] https://pypi.org/project/waitress/
[13] Application programming interface




Obrázek 2.3: Architektura Heroku platformy je rozdělena do několika důležitých segmentů. Patří mezi ně routery, které zaručují správný chod požadavků na aplikaci přes webové stránky, Heroku API (Application Programming Interface) přes které mohou chodit požadavky na server a Dynos, na kterých běží samotná aplikace. Nakonec je zde také Logplex pro logování a různé další přídavné funkcionality. Inspirováno [4]

.

### WSGI

**Web Server Gateway Interface**[14] představuje univerzální rozhraní, pomocí kterého webová aplikace a webový server komunikuje. Aplikace může být napsána v jazyce Python nebo některém Python frameworku. WSGI server přijímá požadavky od klienta, přeposílá je aplikaci a nakonec přepošle odpověď aplikace zpět klientovi. Mezi WSGI HTTP servery patří Gunicorn a Waitress server, které podporují mnoho frameworků včetně Djanga.

[14] http://wsgi.tutorial.codepoint.net/

# Kapitola 3

# Formát PDF

Portable Document Format [10] je formát vyvinutý firmou Adobe. Poskytuje možnost vytvořit dokument s formátovaným textem, obrázky, grafikou atd. Formát je nezávislý na hardwaru, softwaru či platformě a je založený na jazyce PostScript[1].

## 3.1 Objekty

PDF podporuje 5 základních objektů a 3 složené objekty. Všechny tyto objekty mohou být navzájem propojeny pomocí nepřímých odkazů.

- Mezi základní objekty patří:
  - **Celá a reálná čísla:** *Celé číslo* je zapsáno jako jedna nebo více číslic s volitelným znaménkem plus nebo mínus před první číslicí. *Reálné číslo* je zapsáno jako jedna či více číslic, které obsahují tečku oddělující celočíselnou část a zlomek. U reálných čísel je také možnost přidat znaménko mínus nebo plus.
  - **Řetězce:** Skládá se ze série bajtů napsaných mezi kulatými závorkami, například: `(String)`. V bajtech mezi kulatými závorkami se mohou vyskytovat escape sekvence začínající zpětným lomítkem. Řetězec lze také zapsat jako sekvenci hexadecimálních čísel mezi lomenými závorkami: `<537472696e67>`.
  - **Jména:** Jsou používaná jako klíče slovníků. Začínají lomítkem, nesmí obsahovat bílé znaky a rozlišují se velká a malá písmenka: `/Name`.
  - **Boolean hodnoty:** Jedná se o dvě hodnoty, klíčové slova, `true` a `false`.
  - **Objekt null:** Je označen klíčovým slovem `null`.
- Mezi složené objekty patří:
  - **Pole:** Obsahují seřazenou kolekci objektů, včetně dalších polí. Jedná se o heterogenní datovou strukturu, protože všechny položky pole nemusí být stejného typu. Pole jsou zapsány v hranatých závorkách: `[24 11 (String)]`.
  - **Slovníky:** Reprezentují neseřazenou kolekci dvojic klíč-hodnota. Pokud je ve slovníku nalezen daný klíč, je možné získat přiřazenou hodnotu. Klíče jsou objekty typu jméno a hodnoty mohou být jakýkoli typ objektu včetně slovníku nebo null objektu. Slovníky jsou zapsány mezi dvojitými lomenými závorkami: «`/key (value) /secondKey 2`».

[1] https://www.adobe.com/content/dam/acom/en/devnet/actionscript/articles/PLRM.pdf

– **Datové toky:** Datové toky se používají pro vkládání binárních dat. Nacházejí se v objektu s nepřímým odkazem. První položkou je slovník obsahující délku dat a případně další informace. Za ním následuje klíčové slovo značící začátek toku `stream`, nový řádek a následně bajty a ukončovací klíčové slovo `endstream`:

```
5 0 obj
    << /Length 42 >>
    stream
    BT /F1 24 Tf 100 700 Td (Hello World) Tj ET endstream
endobj
```

**Nepřímé odkazy** mají rezervované klíčové slovo `R`. Slouží k tomu, aby se data četla pouze pokud jsou požadována. Obsah souboru PDF je díky nim rozdělen do samostatných objektů. Například nepřímý odkaz na objekt 24 je zapsán takto: `24 0 R`, kde číslo 24 představuje identifikátor objektu a číslo 0 značí číslo generování.

## 3.2 Struktura souboru

Soubor PDF se skládá ze čtyř základních částí, jak znázorněno i na obrázku 3.1:

**Hlavička:** První řádek udává, v jaké verzi je soubor PDF, například `%PDF-1.6` značí že je soubor ve verzi *1.6*. Verze je zpětně kompatibilní. V souboru PDF se často vyskytují i binární data, například ve formě obrázku. Aby při zpracování souboru jasné, že se v něm vyskytují binární data, tak se na druhý řádek hlavičky přidá znak procenta a za něj se přidají alespoň čtyři znaky, které se nevejdou do sedmi bitů (jejich hodnota ASCII kódování[2]je větší než 127), například: `%âãÏÓ`.

**Tělo:** Skládá se ze sekvence objektů. Každý objekt začíná na novém řádku číslem objektu, číslem generace a klíčovým slovem `obj`, tedy `1 0 obj` je objekt s číslem 1 nulté generace. Za obsahem objektu je na novém řádku vloženo klíčové slovo `endobj`.

**Cross-reference tabulka:** Obsahuje tabulku bajtového offsetu (posunu) od začátku dokumentu pro každý objekt z těla dokumentu. Při získávání hodnoty nepřímo odkazovaného objektu, se využívá tato tabulka pro získání pozice objektu (bajtového offsetu) v dokumentu a není tak třeba prohledávat všechny objekty. Každý objekt je v těle identifikován podle svého čísla a čísla generování, které určuje kolikrát je tabulka použita pro jeho generování. Na prvním řádku se nachází klíčové slovo `xref`, na dalším se nachází dvě čísla, první značí číslo prvního objektu a druhé kolik objektů je v těle obsaženo. Následuje obsah tabulky. Na prvním řádku obsahu cross-reference tabulky v obrázku 3.1 můžeme vyčíst, že objekt s číslem 37 začíná na 16 bajtu. Následně lze odvodit z třetího řádku, že objekt s číslem 39 začíná na bajtu 1522.

**Trailer:** Obsahuje metadata o dokumentu. Na prvním řádku se nachází klíčové slovo `trailer`. Následuje slovník, který povinně obsahuje dvě položky: `/Size` a `/Root`. `Size` udává celkový počet záznamů v cross-reference tabulce. `Root` je číslo objektu dokumentového katalogu, tedy hlavního elementu grafu objektů v těle souboru. Po slovníku následuje řádek s klíčovým slovem `startxref`. Za ním následuje řádek s počtem bajtového offsetu od začátku souboru cross-reference tabulky souboru. Je ukončen řádkem značím konec souboru `%%EOF`.

[2] http://www.asciitable.com/

```
%PDF-1.6
37 0 obj <</Linearized 1/L 20597/O 40/E 14115/N 1/T 19795/H [ 1005 215]>>
endobj

xref
37 9
0000000016 00000 n
0000001386 00000 n
0000001522 00000 n
0000001787 00000 n
0000002250 00000 n
0000002274 00000 n
0000002423 00000 n
0000002844 00000 n
0000002888 00000 n

trailer
<</Size 71/Prev 19784/XRefStm 1220/Root 39 0 R/Encrypt 38 0 R/Info 6 0 R>>
startxref
0
%%EOF
```

1: Header

2: Body

3: Cross-Reference Table

4: Trailer

Obrázek 3.1: Struktura jednoduchého jednostránkového souboru PDF. Soubor PDF se skládá ze čtyř základních částí. První částí je hlavička obsahující verzi PDF. Následuje tělo souboru obsahující objekty. Třetí částí je Cross-Reference table – tabulka referencí obsahující bajtový offset nepřímých objektů, slouží k rychlému přístupu k objektům v těle dokumentu. Trailer je poslední částí, obsahuje informace o dokumentu jako celku.[4]

## 3.3 Struktura dokumentu

Formát PDF kromě struktury souboru představuje i strukturu dokumentu – obsah těla souboru, který se skládá z několika povinných sekcí:

**Dokumentový katalog** Katalog je specifikovaný v trailer části souboru – v elementu slovníku `/Root`. Je označený pomocí klíče `/Type` s hodnotou `/Catalog`. Obsahuje odkaz do kořenového objektu stromu stránek pod klíčem `/Pages`. Tento objekt má následující tvar, kde můžeme vidět, že stránky se nachází v objektu `2 0 R`:

```
1 0 obj
  << /Type /Catalog
     /Pages 2 0 R
  >>
endobj
```

**Strom stránek** Jedná se o kořen stromu stránek, obsahuje slovník který je označen klíčem `/Type` s hodnotou `/Pages`. Obsahuje informace o stránkách v souboru, jejich počet pod klíčem `/Count` a nepřímé odkazy v poli v hodnotě klíče `/Kids`. Zde můžeme vidět že dokument má 2 stránky a jsou to objekty 11 a 24:

[4] Inspirováno: https://www.researchgate.net/figure/An-example-of-a-simple-PDF-file-structure-that-consists-of-one-page-that-contains-a_fig1_326102942

```
2 0 obj
    << /Type /Pages
     /Kids [ 11 0 R
             24 0 R]
     /Count 2
    >>
endobj
```

**Stránky** Dokument obsahuje jednu anebo více stránek, informace o stránce jsou uloženy ve slovníku označeným klíčem `/Type` s hodnotou `/Page`. Každá stránka má svoje *zdroje* pod klíčem `/Resources`, které obsahují například použitá písma nebo obrázky, a obsahuje odkaz na rodičovský element pod klíčem `/Parent`. Dále je zde samotný *obsah stránky* pod klíčem `/Contents`, kde nalezneme způsob jakým se má text či jiná grafika na stránce vykreslit. Položka slovníku `/MediaBox` určuje výšku a šířku stránky. Tento objekt může mít následující podobu:

```
24 0 obj
    << /Type /Page
     /MediaBox [0 0 612 792]
     /Resources 3 0 R
     /Parent 2 0 R
     /Contents [4 0 R]
    >>
endobj
```

# Kapitola 4

# Návrh aplikace

Jednou z částí softwarového vývoje je návrh aplikace. V této části je potřeba analyzovat a specifikovat požadavky na produkt. Jaké existují řešení? Pro kterou cílovou skupinu má aplikace sloužit? Jaké jsou požadavky uživatelů na aplikaci? Jak bude aplikace fungovat? Jak bude vypadat uživatelské rozhraní? Na tyto otázky se zaměřuje tato kapitola.

## 4.1 Existující řešení

Řešení které jsem nalezla, buďto počítají znaky pouze z textového vstupu, nebo počítají znaky ze souboru PDF, ale nepřepočítají je na normostrany viz tabulka 4.1. Nástroj který by rozdělil dokument na kapitoly nebo přepočítával na normostrany i obrázky jsem nenalezla.

- **normostrana.cz** Velmi jednoduchý webový nástroj, spočítá znaky vložené do textového pole, znaky následně přepočítá na normostrany.
- **pocetznaku.cz** Tento nástroj také počítá s textem vloženým přímo do textového pole, ale nabízí mimo počtu znaků a normostran další statistiky jako počet mezer, počet slov, počet vět atd.
- **prepostseo.com/word-count-character-count-tool** Nástroj dokáže načítat data ze souboru PDF, DOCX nebo TXT. Je zde také možnost zadat text přímo do textového pole. Výstupem nástroje je počet znaků, slov, vět a předpoklad času, který zabere čtení textu.

| **Webový nástroj** | *normostrana.cz* | *pocetznaku.cz* | *word-count-character-count-tool* | ***Navrhnuté řešení*** |
|---|---|---|---|---|
| **Textový input** | ✓ | ✓ | ✓ | |
| **Znaky z PDF** | | | ✓ | ✓ |
| **Normostrany** | ✓ | ✓ | | ✓ |
| **Statistiky** | ✓ | ✓ | ✓ | ✓ |
| **Obrázky** | | | | ✓ |
| **Kapitoly** | | | | ✓ |

Tabulka 4.1: Porovnání funkcí zkoumaných webových služeb pro analýzu textu a navrhovaného řešení.

## 4.2 Cílová skupina

Aplikace má umožnit počítání rozsahu a získání statistik z textu závěrečných prací. Primárně se tedy zaměřím na tyto dvě skupiny uživatelů:

**Studenti vysoké školy, kteří píší závěrečnou práci:** Studenti mají předem stanovený očekávaný počet normostran, chtějí si průběžně kontrolovat kolik stran již mají napsáno. Potřebují také zjistit kolik normostran představují obrázky a zda nemají příliš velký poměr mezi obrázky a textem. Pro možnost zkvalitnění textu chtějí také zkontrolovat, která slova používají příliš často.

**Vedoucí a oponenti závěrečných prací:** Prověřují jestli student dodržel požadovaný počet normostran. Dále prověřují poměr obrázků a textu - student mohl doplnit potřebný počet normostran nepodstatnými obrázky. Potřebují přehled a velikost obrázků - zda velikost obrázků odpovídá obsahové hodnotě. Jaká slova požíval student nejčastěji - jakou má slovní zásobu, odpovídá to tématu práce?

## 4.3 Požadavky na aplikaci

Jak je naznačeno na obrázku 4.1, webové uživatelské rozhraní aplikace by mělo uživateli umožnit vložit soubor, který se následně nahraje na server. Na serveru se soubor zpracuje a získají se tak statistiky, které se pošlou zpět do rozhraní, kde jsou vizualizovány. Základními požadavky na aplikaci jsou:

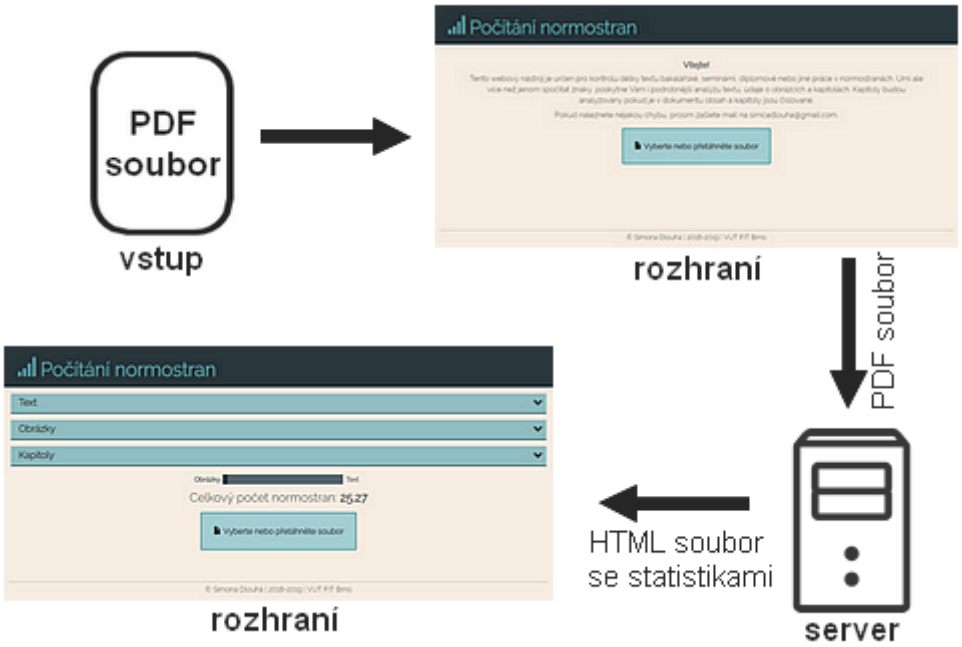
**Jednoduché rozhraní a ovládání:** Uživatelské rozhraní a ovládání je velmi jednoduché a intuitivní. Uživatel tedy ví kam nahrát soubor, jak si zobrazit vybrané statistiky, jak případně nahrát další soubor.

**Formát souboru:** Protože existuje několik možných editorů ve kterých lze psát závěrečnou práci a každý může mít svůj vlastní formát, je potřeba zvolit vhodný formát souboru ze kterého se budou získávat statistiky. Pro tento účel se hodí právě formát PDF, protože jak z LaTeX tak z Microsoft Word dokumentu lze vygenerovat PDF soubor.

**Statistiky z textu:** Nástroj spočítá znaky a následně je přepočítá na normostrany. Dále umí získat i další statistiky jako počet slov, nejčastěji používaná slova nebo počet vět.

**Statistiky z obrázků:** Nástroj poskytuje informace o počtu obrázků v dokumentu, jejich celkovou velikost, průměrnou velikost obrázků a seznam obrázků. Tento seznam obsahuje obrázky seřazené podle velikosti s informací na které stránce dokumentu se nachází. Obrázky jsou přepočítány do normostran, protože se do nich počítají.

**Statistiky z kapitol:** Každá kapitola obsahuje údaje o tom kolik má znaků, normostran a obrázků s jejich velikostmi. Také obsahuje údaj o tom na které stránce začíná a na které stránce končí.




Obrázek 4.1: Princip aplikace – po nahrání souboru PDF přes rozhraní je poslán požadavek na server kde dojde ke zpracování souboru a do rozhraní se v odpovědi vrátí vygenerovaná stránka HTML, zobrazující získané statistiky o souboru PDF.

## 4.4 Zpracování souboru

Soubor se na server pošle pomocí HTTP požadavku metodou POST s nastaveným typem obsahu multipart/form-data. Na serveru se soubor uloží ve formě dočasného souboru.

Na vývoj serverové části bude použit webový framework Django, který typicky pracuje s návrhovým vzorem model-view-controller. Pro tento nástroj není potřeba ukládat data do databáze, nebudou zde tedy žádné modely. Vytvoří se instance třídy `File`, která bude obsahovat vlastnosti představující různé druhy statistik získaných ze souboru. `File` bude obsahovat funkci pro inicializaci z nahraného souboru PDF. Obsah souboru PDF bude získán pomocí knihovny, která umožňuje zpracování souboru po stránkách. Jednotlivé stránky se budou procházet v cyklu, aby bylo možné průběžně vyhodnocovat statistiky obrázků, také zde bude prováděna kontrola a načtení obsahu.

Získané údaje se předají do view, které je následně vloží do HTML šablon uložených na serveru a takto dynamicky vygenerovaná stránka se pošle jako odpověď na daný HTTP požadavek.

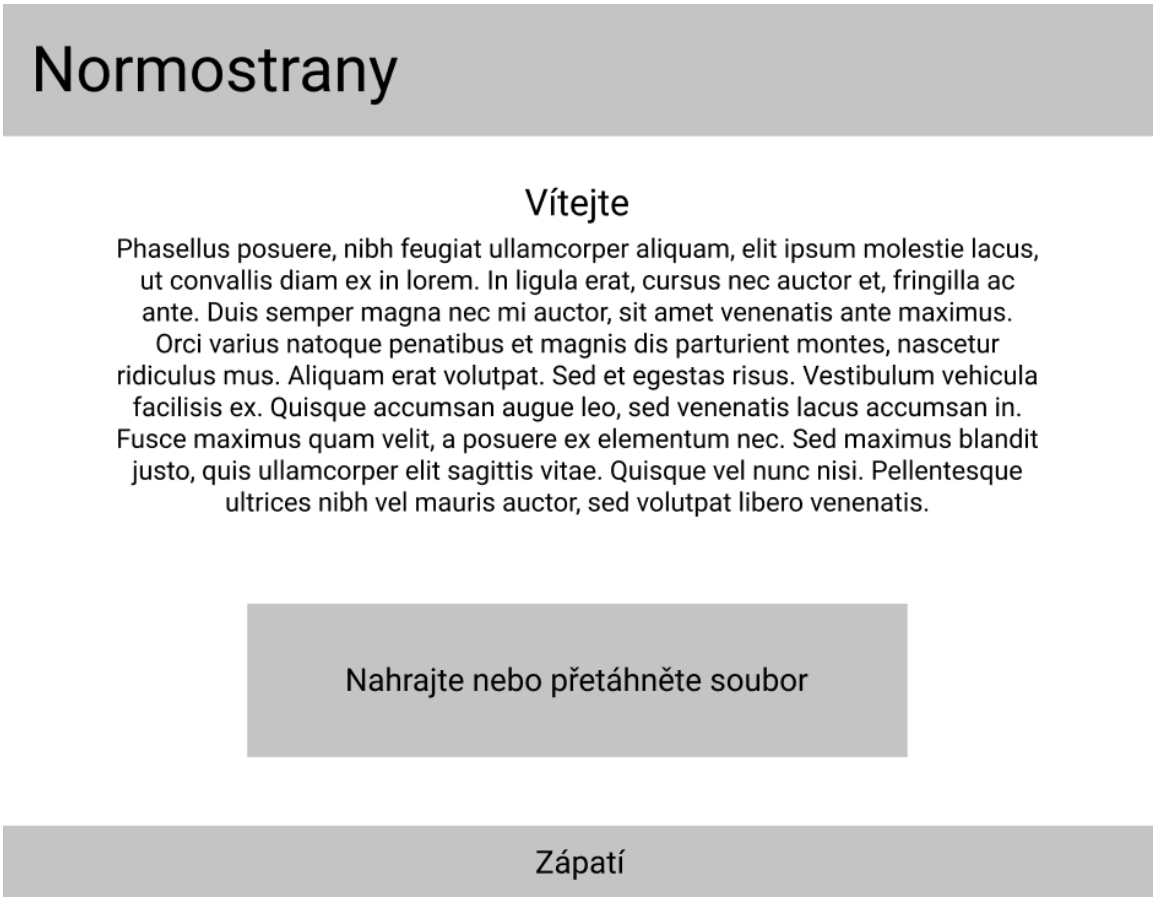
## 4.5 Uživatelské rozhraní

Bude se jednat o webovou aplikaci, která bude obsahovat pouze jedinou stránku zobrazující všechny informace. Tato stránka bude ve dvou módech:

### Hlavní stránka před nahráním obrázku

Jak lze vidět na obrázku 4.2, po prvním vstupu na stránku bude zobrazen úvodní text. Pod úvodním textem se bude nacházet tlačítko, které bude dominovat stránce. Klíčovými prvky jsou:

- **Úvodní text:** Bude obsahovat nápovědu, způsob použití nástroje, kontakt na administrátora.
- **Tlačítko pro nahrání souboru:** Po kliknutí na toto tlačítko se zobrazí výběr souboru, ve kterém bude výběr omezen na formát souboru PDF. Po výběru souboru se automaticky odešle na server požadavek ke zpracování. Pokud uživatel vybere špatný typ souboru, zobrazí se chybová hláška. Pokud se nahraje soubor formátu PDF, přejde stránka do módu po nahrání obrázku. Toto tlačítko disponuje velkou plochou, protože po přetáhnutí souboru se vykoná stejná logika jako po výběru souboru – funkce Drag&Drop.



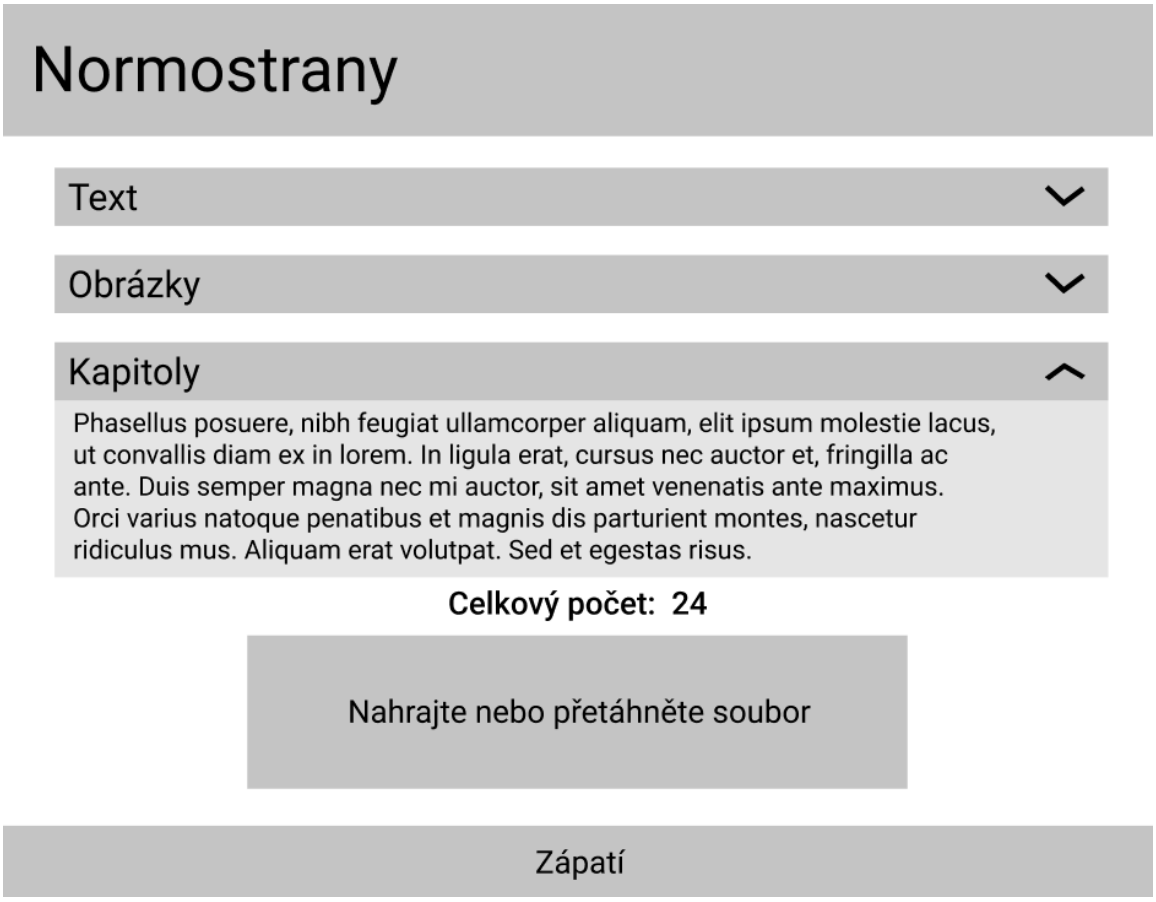

Obrázek 4.2: Návrh hlavní stránky před nahráním obrázku, klíčovými prvky jsou úvodní text a tlačítko pro výběr souboru.

### Hlavní strana po nahrání obrázku

Po nahrání a zpracování souboru se zobrazí získané statistiky jak lze vidět na obrázku 4.3. Budou zde tři rozšířitelné sekce *Text*, *Obrázky* a *Kapitoly*. Pod těmito sekcemi se bude na-

cházet celkový počet normostran. Dále zde bude stejné tlačítko jako v módu před nahráním obrázku, sloužící pro analýzu dalšího dokumentu. Klíčovými prvky tedy jsou:

- **Rozšiřitelné sekce:** Po kliknutí se zobrazí nebo skryje detail sekce (tzv. funkce toggle) a změní se směr šipky značící rozbalení nebo zabalení sekce.
- **Detaily rozšířitelných sekcí:** Zde budou zobrazeny statistiky týkající se dané sekce: grafy, tabulky, číselné údaje. Údaj bude jasně pojmenovaný.
- **Celkový počet normostran**
- **Tlačítko pro nahrání souboru:** Bude mít stejnou logiku jako tlačítko v druhém módu.




Obrázek 4.3: Návrh hlavní stránky po nahrání obrázku, klíčovými prvky jsou rozšířitelné sekce, celkový počet normostran a tlačítko pro výběr souboru.

# Kapitola 5

# Implementace

Tato kapitola se zabývá výslednou implementací nástroje. Budou zde popsány datové struktury použité při zpracování souboru, vyvinuté algoritmy pro zpracování textu, obrázků a kapitol. Shrnuje také výsledné uživatelské rozhraní těchto sekcí a technologie použité k implementaci. Nástroj byl průběžně zveřejňován přes cloudovou platformu Heroku na stránce http://standardpages.herokuapp.com/standardpages/.

## 5.1 Vystavování na server

Aplikace byla vystavovaná na platformu Heroku pomocí GITu. Pro běh aplikace je nutné zprovoznit také web server, který zpracovává HTTP požadavky. Nejdříve byl použit web server **gunicorn**[1], který Heroku doporučuje pro Django aplikace. Při pomalém internetovém připojení však docházelo k tomu, že nahrání souboru a zpracování trvalo příliš dlouho. Odpověď na požadavek byla tedy zaslána pozdě a došlo k timeoutu (vypršení platnosti požadavku), webová aplikace zobrazovala chybu „Application Error“. V záznamech činnosti na serveru byla zaznamená chyba „Request Timeout“, způsobená nastavením Heroku, to stanovuje maximální dobu požadavku na 30 sekund, kterou nelze prodloužit. Jedním z možných řešení byla změna serveru[2] na server **waitress**[3]. Po změně na tento server, který používá bufferovaní požadavků a odpovědí, již k tomuto problému nedochází.

Pro správný běh aplikace je potřeba nainstalovat také balíčky, které se v implementaci využívají. Závislosti[4], které jsou potřeba pro běh aplikace jsou uvedeny v souboru *requirements.txt* v kořenovém adresáři aplikace. Po vystavení na Heroku jsou balíčky uvedené v tomto souboru automaticky nainstalovány.

## 5.2 Datové struktury

V rámci implementace bylo vytvořeno několik tříd pro reprezentaci souboru a jeho částí. Diagram těchto tříd je modelován na obrázku 5.1.

**Třída rectangle** Tato třída je použita při zpracování obrázků. Obrázky jsou uloženy jako kolekce instancí třídy rectangle, slouží například pro získání rozměrů obrázku.

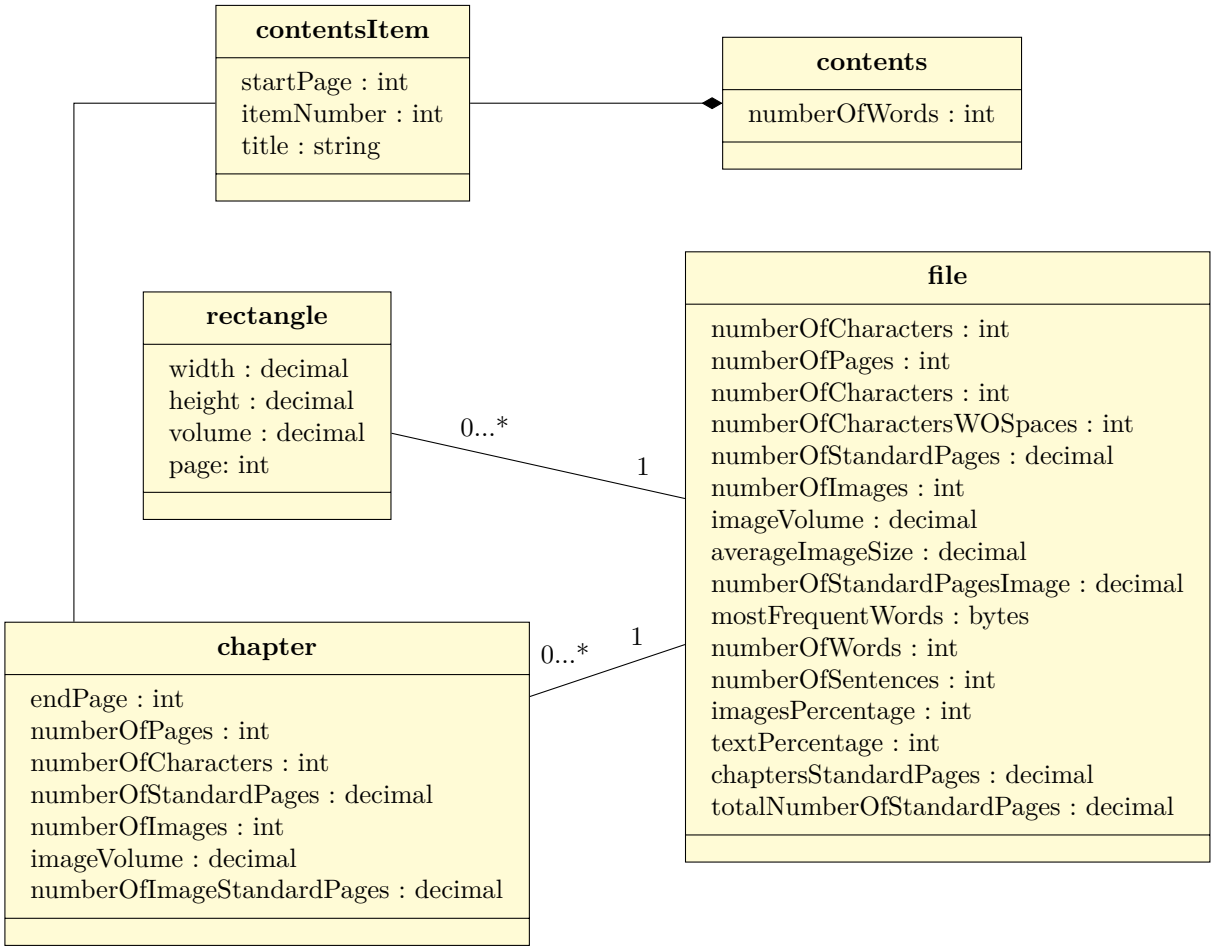
[1] https://gunicorn.org/
[2] https://blog.etianen.com/blog/2014/01/19/gunicorn-heroku-django/
[3] https://pypi.org/project/waitress/
[4] https://devcenter.heroku.com/articles/python-pip

**Třída contentsItem** Slouží pro reprezentaci jednotlivých položek ze stránky s obsahem ve zpracovávaném souboru PDF. Po zpracování kapitol je instance této třídy uložená v odpovídající kapitole.

**Třída contents** Dokument vložený do aplikace má často stránku s obsahem dokumentu, třída contents reprezentuje právě zmíněný obsah dokumentu. Obsahuje kolekci instancí třídy `contentsItems` a využívá se při parsování kapitol.

**Třída chapter** Třída chapter reprezentuje kapitolu dokumentu. Všechny kapitoly jsou kolekcí této třídy. Obsahuje atribut `contentsItem`, který představuje odpovídající položku obsahu.

**Třída file** Je zpracovanou reprezentací nahraného souboru PDF, jejíž atributy představují statistické údaje o daném souboru. Obsahuje kolekci instancí třídy chapter v atributu `chapters` – kapitoly extrahované ze zpracovaného textu. Dále je jejím atributem `sortedImages`, což je kolekce instancí třídy rectangle, ta představuje obrázky seřazené od největšího po nejmenší.




Obrázek 5.1: Diagram tříd zobrazující třídy, jejich atributy a vazby mezi třídami.

## 5.3 Hlavní stránka

Ve výsledném řešení je uživatelské rozhraní hlavní stránky obohaceno o několik prvků. Na hlavní stránce po nahrání validního souboru došlo k přidání dvou prvků. Mód lze vidět na obrázku 5.2, nachází se zde navíc název souboru a ukazovátko pro zobrazení poměru obrázků a textu. Pokud se nahraje nevalidní soubor, zobrazí se chybová hláška jak je zobrazeno na obrázku 5.3.

Zpracování souboru probíhá stejně jak bylo navrhnuto v kapitole 4.4. Na zpracování textu ze souboru byla původně použita knihovny pdfminer[5] a obrázky byly získávány knihovnou PyMuPDF. Z důvodu zefektivnění řešení (procházení stránek v jednom cyklu) a také proto, že knihovna PyMuPDF umí extrahovat i text, byla nakonec použita pouze tato knihovna. Knihovna pdfminer se totiž specializuje výhradně na extrahovaní textových dat.

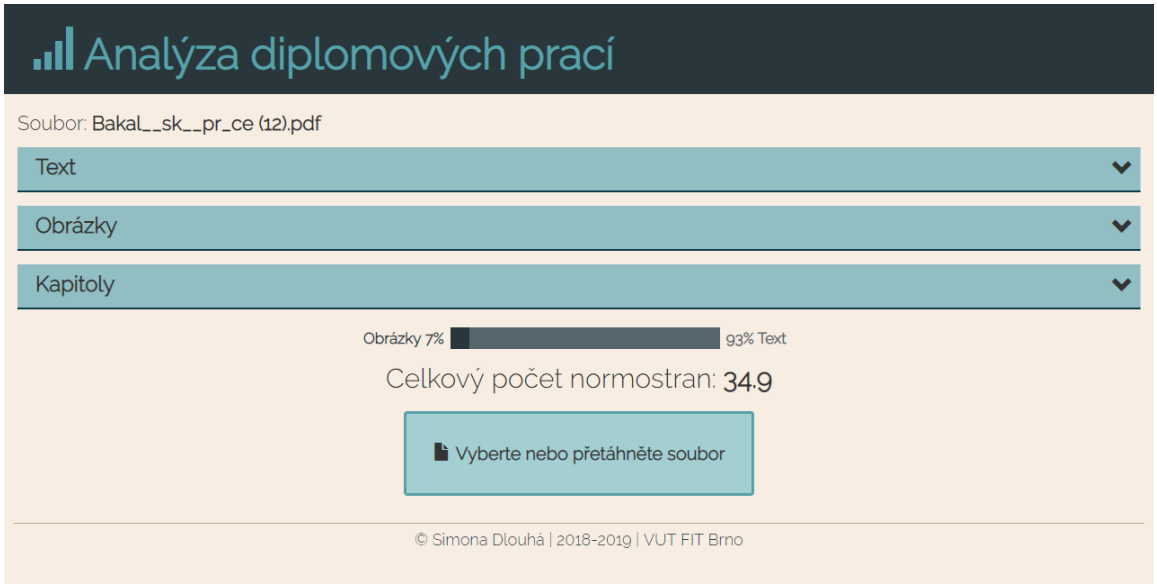
**Název souboru** Uveden pro lepší orientaci uživatele. Značí který soubor uživatel nahrál. Je získán z názvu nahraného souboru.

**Ukazovátko poměru** Funguje na tomto principu: Instance třídy `file` obsahuje atributy `imagesPercentage` a `textPercentage`. Tyto údaje jsou přes view vloženy do HTML šablony, která obsahuje dva HTML elementy div pro zobrazení poměru, vnořené do dalšího HTML elementu div, jejichž šířka je dána procentuální hodnotou, stanovenou těmito atributy. Šířky elementů se nastavují pomocí CSS. Atribut `imagesPercentage` je procentuální zastoupení normostran získaných z obrázků z celkového počtu normostran. Atribut `textPercentage` je procentuální zastoupení normostran z textu z celkového počtu normostran.

**Rozšířitelné sekce** Jsou implementovány pomocí JavaScriptu a CSS. Sekce má při prvním vykreslení nastavené zobrazení obsahu v CSS na `display: none`, to znamená, že obsah sekce je skrytý. Po prvním kliknutí na hlavičku se sekce rozšíří pomocí JavaSriptu – její obsah se stane viditelným přidáním CSS třídy `toggle` definované ve stylovém předpisu, která nastaví zobrazení na blokové `display: block`. Dále se změní směr ikonky šipky. V CSS jsou pro ikonku definované dvě třídy `closed` a `opened`, ve výchozím stavu má ikonka třídu `closed` kde je nastavena transformace na rotaci o 0 stupňů. Po kliknutí se přidá třída `opened` a `closed` se odebere, tato třída má nastavenou rotaci o 180 stupňů. Šipka poté nesměřuje dolů ale nahoru, značící možnost zabalení. Po další kliknutí se sekce zabalí – odebere se třída `toggle` z obsahu a šipka se opět otočí výměnou třídy `opened` za `closed`.

**Tlačítko pro nahrání souboru** Stránka obsahuje formulář pro nahrání souboru, který odesílá data typu „multipart/form-data“ metodou HTTP POST. Tento formulář obsahuje vstup pro výběr souboru a tlačítko pro odeslání formuláře. Tlačítka pro odeslání a výběr souboru jsou vizuálně spojeny v jedno tlačítko, protože jedinou položkou formuláře je právě výběr souboru. Tohoto vizuálního spojení tlačítek je docíleno pomocí JavaScritu a CSS. Po kliknutí na tlačítko se nejprve otevře formulář pro výběr souboru a po výběru souboru se automaticky odešle formulář. Oblast ve které se nachází tlačítko obsahuje handlery v atributech `ondrop` a `ondragover` pro Drag&Drop funkcionalitu, která umožňuje nahrát soubor přetažením na pozici tlačítka. Handlery jsou definované ve skriptu v jazyce JavaScript.

[5] https://pypi.org/project/pdfminer.six/




Obrázek 5.2: Hlavní stránka s nahraným souborem. Nahoře se nachází název nahraného souboru. Pod názvem jsou rozklikávací sekce. Pod sekcemi je ukazovátko poměru obrázků a textu. Je zde i celkový počet normostran a tlačítko pro nahrání souboru.


Obrázek 5.3: Hlavní stránka po nahrání nevalidního souboru. Zobrazuje se chybová hláška s důvodem proč se nahrání souboru nezdařilo. Nad chybovou hláškou je stále vidět úvodní text a pod ní se nachází tlačítko pro opětovné nahrání souboru.

## 5.4 Sekce text

Text práce je jedna z nejpodstatnějších částí analyzovaného dokumentu. Odhadnout počet normostran v dokumentu pocházejících z textu pouhým pohledem není snadné, proto je vhodné použít nástroj, který tento údaj vypočítá. Z toho důvodu je v implementaci nástroje zahrnuta sekce text, která zobrazuje mimo počet normostran také podrobnější údaje o textové náplni práce, například počet znaků a slov.

### Extrakce textu

Text je z dokumentu PDF extrahován pomocí knihovny PyMuPDF. Po extrakci z dokumentu PDF se v získaném textu mohou nacházet některé znaky navíc. Pomocí knihovny re je provedena korekce. Chyby se v textu vyhledávají pomocí regulárních výrazů, uvedených níže. Při nalezení chyby je text upraven tak, aby počet co nejlépe odpovídal reálnému počtu znaků.

- U dokumentu vytvořeného v Microsoft Wordu i u dokumentů z LaTeX, dochází k extrahování i znaků nových řádků. Pro odstranění nových řádků je použit tento regulární výraz: `(\r\n|\r|\n)( |)`
- V LaTeX dokumentu se znaky s diakritikou extrahují jako dva znaky a to základní znak a diakritika. Jak u dokumentů z Microsoft Wordu tak z dokumentů z LaTeX dochází k zalamování slov, tedy přidání pomlčky a mezery. Pro odstranění háčků, čárek, kroužků a zalomení textu jsou použity následující regulární výrazy:
  - `[˘-´]( |)`
  - `U[˚]( |)|[˚] U`
  - `u[˚]( |)|[˚] u`

### Statistické údaje z textu

Výsledné grafické uživatelské rozhraní této sekce je zobrazeno na obrázku 5.4.

**Počet znaků**, atribut `file.numberOfCharacters`, se získá jako délka textu extrahovaného pomocí knihovny PyMuPDF. **Počet znaků bez mezer**, který se nachází v atributu `file.numberOfCharactersWOSpaces`, je délka extrahovaného textu, kde jsou pomocí knihovny re a regex výrazu `text = re.sub(" ", "", text)` odstraněny mezery. Výsledná hodnota znaků bez mezer je délka upraveného textu. Tyto dva statistické údaje jsou užitečné pro porovnání, jestli v textu není příliš mnoho mezer.

Získání **počtu slov** (`file.numberOfWords`) je docíleno díky metodě pro rozdělení řetězce. Tato metoda rozdělí řetězec do seznamu na základě separátoru, výchozí hodnota separátoru je libovolný bílý znak. Počet slov je délka seznamu, který tato metoda vrátí: `len(text.split())`.

**Nejčastěji používaná slova** (`file.mostFrequentWords`) je pole 50-ti nejčastěji používaných slov v dokumentu zobrazených pomocí grafu. Graf je reprezentován obrázkem ve formátu PNG. Nejčastěji používaná slova se získají pomocí algoritmu 1. Slova z extrahovaného textu se postupně prochází a pokud splňují podmínku, že jsou delší než 2 znaky, zaznamenávají se do slovníku. V případě, že se slovo již ve slovníku nachází, zvýší se hodnota jeho záznamu ve slovníku o 1 (zvýší se počet výskytů). Pokud se ve slovníku nenachází, nastaví se jeho hodnota na 1 (první výskyt). Nejčastěji používaná slova jsou vizualizovaná

pomocí *knihovny matplotlib*. Ta interpretuje pole hodnot ze slovníku do obrázku histogramu ve formátu PNG. Tento obrázek se zakóduje do kódování base64[6] pomocí knihovny base64[7] a poté se přes view vloží do šablony HTML. Tento histogram může být použit například k následujícím účelům:

- Úprava výsledného textu – pokud se některé slovo vyskytuje příliš často, lze ho cíleně omezit a nahradit.
- Kontrola, že nejčastější slova jsou k tématu práce, mělo by se pravděpodobně jednat o termíny úzce související s tématem práce.

**Algorithm 1** Nejčastěji používaná slova

```
function GetMostFrequentWords
    text ← extrahovaný text
    wordsDictionary ← prázdný slovník pro slova
    for word in text do
        if len(word) > 2 then
            if word in wordsDictionary then
                wordsDictionary[word] ← wordsDictionary[word] + 1.
            else
                wordsDictionary[word] ← 1.
return sorted(wordsDictionary)
```

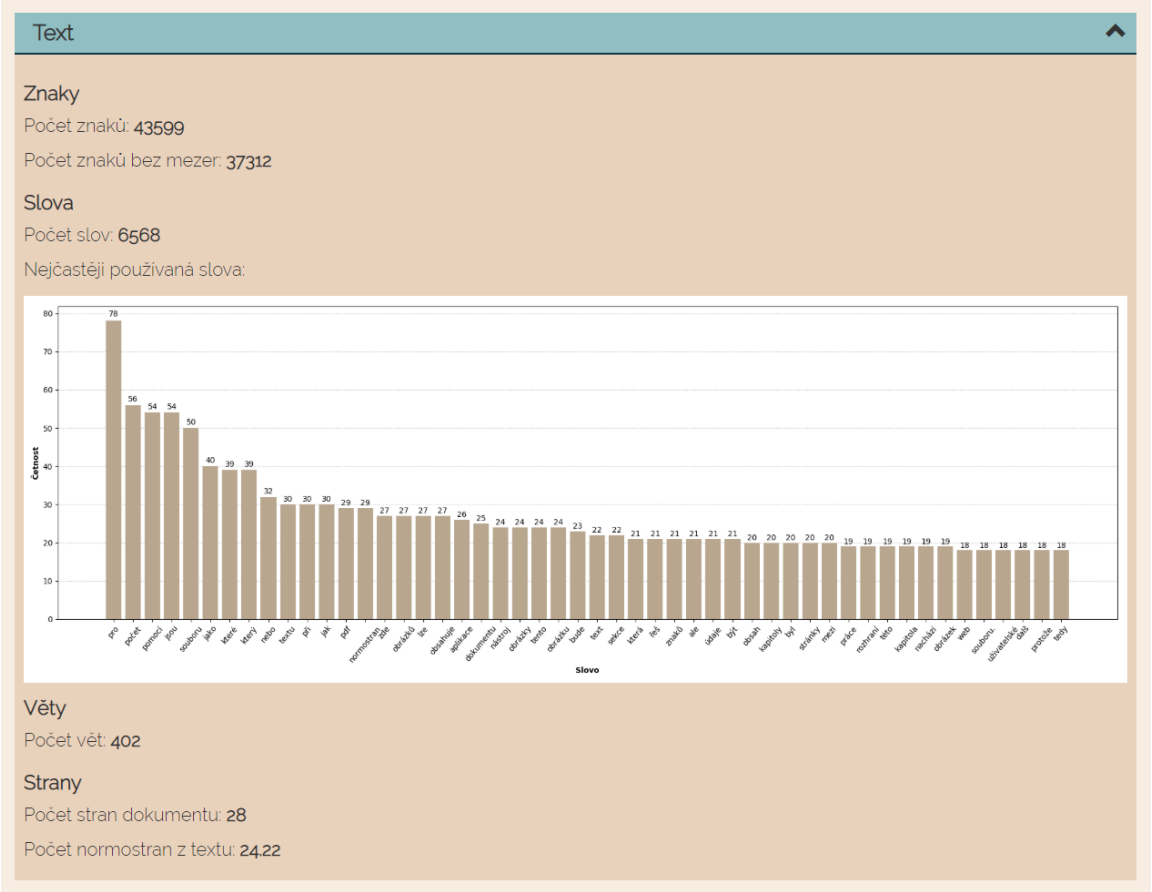
**Počet vět** je získán pomocí knihovny re. Je zde použit regulární výraz, který předpokládá, že věta začíná na velké písmeno a končí na jeden ze znaků z množiny ukončovacích znaků: `([A-Z][^\.!?]*[\.!?])`. Nejdříve se zkompiluje filtr obsahující regulární výraz a poté se na filtru zavolá metoda `findall`, která vyhledá všechny řetězce odpovídající danému regulárnímu výrazu ve filtru. Metoda tedy vrátí seznam všech vět. Počet vět (`file.numberOfSentences`) je poté délka seznamu.

Při procházení dokumentem po stránkách je spočítán **počet stran** analyzovaného dokumentu (`file.numberOfPages`). **Počet normostran z textu**je spočítán z rovnice

$$N = \frac{Z}{1800}, \tag{5.1}$$

kde $Z$ představuje počet znaků včetně mezer a $N$ atribut `file.numberOfStandardPages`. Tento údaj sděluje uživateli kolik normostran má z celkového textu, včetně nadpisů, popisků a citací.

[6] https://blogs.oracle.com/rammenon/base64-explained
[7] https://docs.python.org/2/library/base64.html




Obrázek 5.4: Detail rozšířitelné sekce – Text. Po kliknutí a rozšíření této sekce jsou zde 4 podsekce: Znaky, Slova, Věty a Strany. Sekce obsahuje statistické údaje o textu.

## 5.5 Sekce obrázky

Obrázky se stejně jako znaky počítají do rozsahu normostran práce. Odhad kolik stran obrázek představuje je dán tím, kolik textu by přibližně představoval. Z tohoto důvodu byl navržen mechanismus, který tento odhad pomůže určit. K zobrazení získaných údajů o obrázcích slouží sekce obrázky.

### Extrakce obrázků

Obrázky jsou získány pomocí knihovny PyMuPDF, ale ne pomocí metody kterou pro extrakci obrázků nabízí. Po použití metody co knihovna nabízí pro extrakci obrázků byly sice získány obrázky, ale jejich rozměr byl pouze v pixelech. Rozměr v pixelech navíc neměl souvislost s velikostí stránky, jednalo se o původní původní velikost obrázku. Nebylo tak možné určit jakou oblast stránky obrázek zabírá.

Nejdříve byl obsah stránky extrahován pomocí metody `page.getText("rawdict")` jako slovník. Tento slovník obsahuje bloky s objekty PDF v atributu `blocks`. Extrakce obrázků v blocích probíhá následovně:

1. Bloky se prochází v cyklu a pokud blok obsahuje klíč `image`, nachází se v něm obrázek.

2. Velikost obrázku vůči stránce se určí pomocí atributu `bbox`.

3. Atribut `bbox` je pole o 4 prvcích, které určují umístění obrázku na stránce. Lze tak určit velikost v pixelech na stránce $x_{px}$ a $y_{px}$ a rozšířit tak seznam obrázků.

Přístup pomocí metody `page.getText("rawdict")` vracející slovník obsahující blokové objekty včetně obrázků způsoboval po vystavení na server výrazné problémy s pamětí a časem zpracování, pokud soubor obsahoval velké obrázky. Je to problém z důvodu omezení Heroku na 512 MB paměti a 30-ti sekundového limitu na zpracování požadavku. Příčinou nízké efektivity bylo to, že součástí struktury blokového objektu byl i kompletní obrázek zakódovaný v base64, ten je ovšem pro tento nástroj nepodstatný, je potřeba znát pouze rozměry obrázku. Zvolila jsem tedy jiný přístup. Obsah stránky je extrahován pomocí metody `page.getTextBlocks(True)`, která vrací seznam bloků z aktuální stránky. Parametr `True` značí, že výsledek metody obsahuje také bloky obrázků, které se na stránce nachází, ale bez jejich reprezentace v base64. Bloky se procházejí v cyklu a pokud mají hodnotu typu rovnou hodnotě „1“, jsou zpracovány jako obrázky, jinak jsou ignorovány, protože se jedná o bloky textu. Velikost obrázku vůči aktuální stránce je určena stejně jako v bodech 2. a 3., s tím rozdílem, že na vstupu je místo slovníku seznam a nepoužívá se zde atribut `bbox`, ale informace o umístění na stránce jsou uloženy přímo jako atributy položky tohoto seznamu. Tento přístup je velice přínosný pro rychlost zpracování jednotlivých požadavků a také spotřebovává mnohem méně paměti.

Velikost obrázku v pixelech $x_{px}$ a $y_{px}$ je určená při extrakci obrázků, ale pro získání jeho rozměru v centimetrech je nutné nejdříve určit konstanty pro přepočet pixelů na centimetry. Velikost stránky v pixelech $X$ a $Y$ se nalézá v atributu stránky `MediaBoxSize`. Za předpokladu, že se diplomové práce tisknou na A4[8], jak je naznačeno na obrázků 5.5, se konstanta pro přepočet pixelů na centimetry na straně x $K_x$ spočítají jako

$$K_x = \frac{X}{21} \tag{5.2}$$

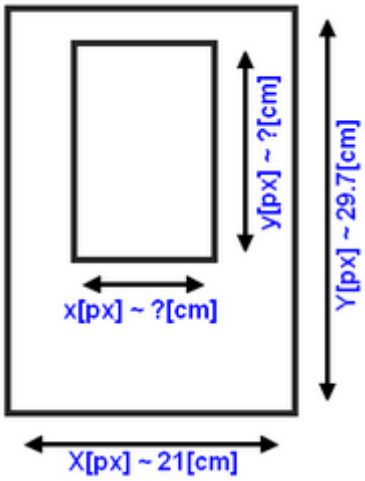
a konstanta pro přepočet pixelů na centimetry na straně y $K_y$ jako

$$K_y = \frac{Y}{29.7}. \tag{5.3}$$

Posledním krokem je přepočítat obrázek na centimetry, tedy vypočítat šířku $x_{cm}$ a výšku $y_{cm}$ obrázku, jedná se o atributy `rectangle.width` a `rectangle.height`,

$$x_{cm} = \frac{x_{px}}{K_x},\ y_{cm} = \frac{y_{px}}{K_y}. \tag{5.4}$$

[8] https://cs.wikipedia.org/wiki/Form%C3%A1t_pap%C3%ADru




Obrázek 5.5: Princip přepočítání velikosti obrázku na stránce v pixelech na centimetry. Stránka o šířce $X$ pixelů a výšce $Y$ se zobrazuje na stránku A4. Poměrem velikosti stránky v pixelech a v centimetrech lze získat poměry podle kterých se zobrazují i strany obrázku $x$ a $y$.

Obrázky

Počet obrázků: **13**

Počet cm$^2$: **657.9**

Průměrná velikost obrázku (v cm): **7.11×7.11**

Počet normostran z obrázků: **1.05**

| Obsah [cm$^2$] | Výška [cm] | Šířka [cm] | Stránka |
|---|---|---|---|
| 126.6 | 8.44 | 15.0 | 15 |
| 126.6 | 8.44 | 15.0 | 19 |
| 122.55 | 8.17 | 15.0 | 16 |
| 65.9 | 6.59 | 10.0 | 14 |
| 46.1 | 4.61 | 10.0 | 11 |
| 31.4 | 6.28 | 5.0 | 17 |
| 31.05 | 6.21 | 5.0 | 18 |
| 23.7 | 1.58 | 15.0 | 15 |
| 21.9 | 4.38 | 5.0 | 21 |
| 16.05 | 3.52 | 4.56 | 17 |
| 16.05 | 3.52 | 4.56 | 17 |
| 16.05 | 3.52 | 4.56 | 17 |
| 13.95 | 2.79 | 5.0 | 20 |

Obrázek 5.6: Detail rozšířitelné sekce – Obrázky. V této sekci se po kliknutí a rozbalení nachází statistické údaje jako počet obrázků, plocha kterou tyto obrázky zabírají, průměrná velikost obrázku, počet normostran z obrázků a tabulka s obrázky od největšího po nejmenší.

### Statistické údaje z obrázků

Získané statistické údaje o obrázcích a jejich vizualizaci v uživatelském rozhraní je možné vidět na obrázku 5.6.

**Počet obrázků** `file.numberOfImages` je počet položek získaného seznamu obrázků. Celková plocha, kterou by obrázky zabíraly na stránce o velikosti A4 se nachází v atributu `file.imageVolume`. **Počet cm$^2$** je získán sečtením atributů `rectangle.volume` všech položek seznamu obrázků. **Průměrná velikost** strany obrázku na stránce A4 je uložena v atributu `file.averageImageSize` a je počítána jako

$$\sqrt{\frac{V}{N}}, \tag{5.5}$$

kde $V$ představuje atribut `file.imageVolume` a $N$ `file.numberOfImages`.

**Počet normostran z obrázků**, atribut `file.numberOfStandardPagesImage`, prakticky udává počet stránek, které obrázky zabírají na stránkách A4 bez okrajů. Velikost okrajů byla určena jako průměrná hodnota z doporučení od různých škol (například FIT VUT[9] nebo PedF UK[10]) a to 5.5 cm na šírku a 7.2 cm na výšku. Tento odhad přibližně odpovídal zpětné vazbě od uživatelů. Počet normostran $N$ je dán poměrem `file.imageVolume` $V$ a obsahem stránky A4 bez okrajů, tedy

$$N = \frac{V}{15.5 * 22.5}. \tag{5.6}$$

**Tabulka s obrázky** obsahuje obrázky seřazené od největšího po nejmenší a její položky jsou uloženy v atributu `file.sortedImages`. Každý obrázek zde má šířku, výšku, obsah a stránku na které se nachází. Tato tabulka může mít využití spíše pro učitele, kteří se mohou podle čísla stránky v dokumentu podívat na to, jestli největší obrázky dávají smysl a nejedná se pouze o obrázky vyplňující místo.

## 5.6 Sekce kapitoly

Kapitoly rozdělují diplomovou práci na logické celky. Autora práce nebo profesora hodnotícího práci zajímá, jaké mají jednotlivé kapitoly složení a jaký mají rozsah. Dále mohou pozorovat, jaký rozsah má daná kapitola oproti ostatním kapitolám. K tomu slouží sekce kapitoly, která zobrazuje informace o kapitolách.

### Extrakce kapitol

Pro extrakci kapitol jsem vyzkoušela několik přístupů. V prvním se kapitoly extrahují na základě velikosti písma, další z dokumentu extrahuje kapitoly za předpokladu číslovaných kapitol a poslední nejdříve získá obsah a podle něho se orientuje v dokumentu. Nejúspěšnější byl poslední přístup, který jsem nakonec použila ve výsledném řešení.

[9] http://www.fit.vutbr.cz/info/szz/pokyny_bp.php#pozadavky
[10] http://it.pedf.cuni.cz/metodika/?kap=3

**Algorithm 2** Algoritmus pro získání obsahu

```
function GetContents(text)
    contents ← prázdná fronta
    itemNumber ← 0
    title ← prázdný řetězec
    pageNumber ← 0
    for word in text do
        if int(word) then
            if not(itemNumber) then
                itemNumber ← word
            else
                pageNumber ← word
        else
            title ← title + word.
        if itemNumber and pageNumber and notEmpty(title then
            contents.put(ContentsItem())
            itemNumber ← 0
            title ← prázdný řetězec
            pageNumber ← 0
return contents
```

První z přístupů je založený na tom, že nadpisy a text mají rozdílné **velikosti písma**. Lze předpokládat, že nadpis kapitoly první úrovně má největší velikost, nadpis druhé úrovně je menší než nadpis první úrovně a zároveň je větší než text v odstavcích. Při implementaci ale došlo k tomu, že knihovna PyMuPDF při extrakci textu do dokumentu HTML pomocí metody `page.getText("html")`, který obsahuje informace o použitém fontu, extrahovala každý znak zvlášť a ne text či titulek jako celek. Bylo by tedy výpočetně náročné zjistit jakým písmem je napsaná určitá část textu. Dalším problémem bylo určit přesně hierarchii velikostí písma, největší použité písmo nemusí znamenat titulek první úrovně, protože větší písmo může být použito v úvodní straně a naopak nejmenší velikost písma nemusí znamenat text, protože se může jednat například o popisek obrázku.

Princip **číslovaných kapitol** je založený na předpokladu, že nadpis kapitoly je v následujícím formátu: *<Číslo kapitoly> <Titulek> <Znak nového řádku>*. Obsah kapitoly se poté nachází mezi dvěma nadpisy nebo nadpisem a koncem dokumentu. Problém spočívá v tom, že v textu se mohou nacházet čísla a za nimi následuje text, který končí koncem řádku i mimo názvů kapitol Například v odstavci *„Obsahuje 24 položek seznamu.“* by byl detekován nadpis kapitoly číslo 24 jako „položek seznamu.“. Tomu se dá částečně zamezit ověřením, jestli číslo kapitoly odpovídá posloupnosti číslování kapitol. Hlavní problém je ale se stránkou s obsahem, která je povinnou částí diplomové práce[11]. Formát řádků v obsahu je tímto algoritmem většinou detekován jako začátky kapitol.

Na základě předchozího algoritmu byl navržen výsledný, s tím rozdílem, že se nejdříve získá obsah dokumentu. Způsob získání obsahu je popsán v algoritmu 2. U **položky obsahu** se očekává tento formát: *<Číslo kapitoly> <Titulek> <Číslo stránky>*. Kvůli tomuto formátu algoritmus nebude fungovat správně pokud se bude v titulku nacházet číslo, což ale není běžné a také pokud nebudou kapitoly označeny číslem. Ukončující podmínkou tohoto algoritmu je buďto konec souboru nebo slovo neodpovídající formátu.

[11] http://www.fit.vutbr.cz/info/szz/pokyny_bp.php.cs

**Kapitoly** jsou získány pomocí algoritmu 3. Pokud je nalezeno slovo značící obsah, proběhne pokus o získání obsahu. Po získání obsahu se v dokumentu kapitoly detekují pomocí položek obsahu a počítají se znaky mezi jednotlivými kapitolami.

**Algorithm 3** Algoritmus pro získání kapitol

```
1:  function GetChapters
2:      text ← extrahovaný text
3:      contents ← prázdná fronta
4:      chapters ← prázdný seznam
5:      numberOfCharacters ← 0
6:      i ← 0
7:      while i < len(text) do
8:          word ← text[i]
9:          if not(contents) then
10:             if word == "Obsah" then
11:                 contents ← GetContents(text[i:]))
12:                 if contents then
13:                     contentsItem = contents.get()
14:         else
15:             if word == contentsItem.itemNumber then
16:                 if contentsItem.title == text[i:title.len] then
17:                     chapters.append(Chapter())
18:                     numberOfCharacters ← 0
19:                     i ← i + title.len
20:                     contentsItem = contents.get()
21:             numberOfCharacters ← numberOfCharacters + len(word)
22:         i ← i + 1
23:     if contents then
24:         chapters.append(Chapter())
25: return chapters
```

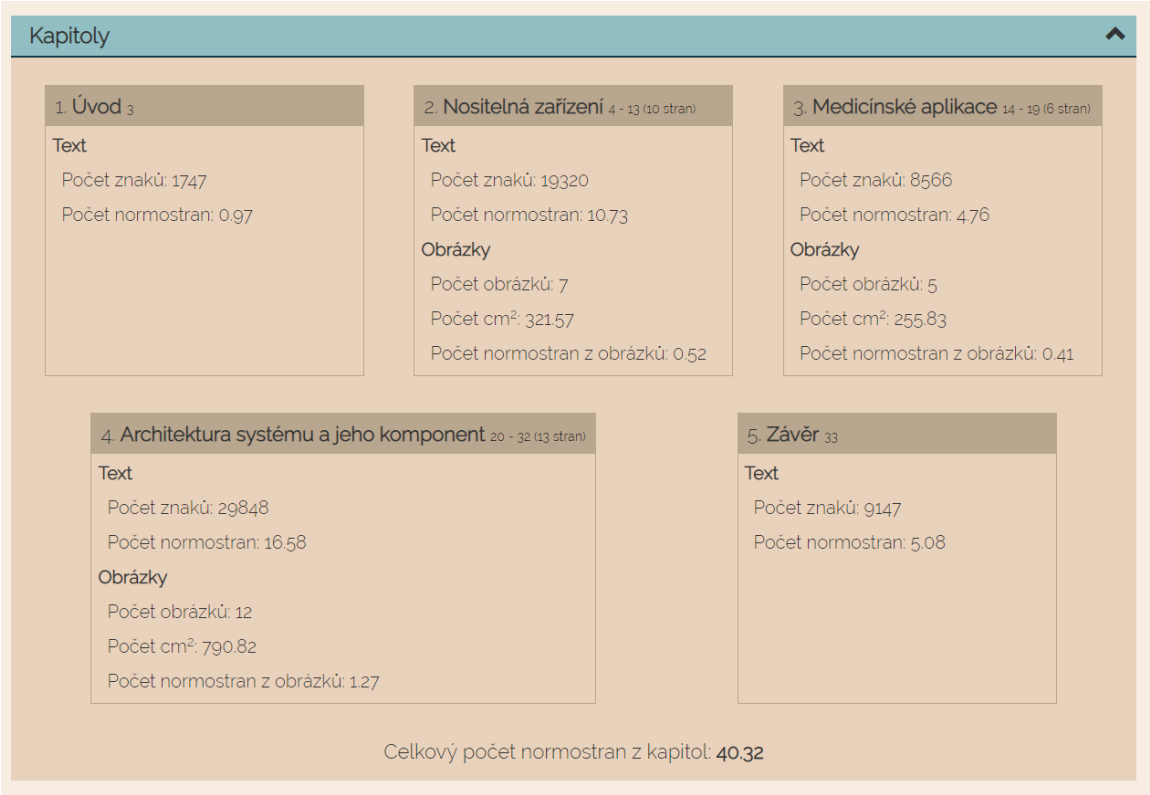
### Statistické údaje z kapitol

Údaje o kapitolách jsou v uživatelském rozhraní zobrazeny na obrázku 5.7. Každá kapitola je zobrazena v bloku, který obsahuje údaje rozdělené na textové a obrázkové statistiky. Na konci se nachází celkový počet normostran z kapitol. Každá ze získaných kapitol obsahuje **rozsah stránek** v atributu `chapter.numberOfPages`, který je získán jako stránka, na které podle obsahu kapitola začíná (`chapter.ContentItem.startPage`) do stránky před kterou začíná následující kapitola (`chapter.endPage`). Tento údaj ukazuje, jaký rozsah stránek dokumentu má kapitola a lze ho porovnat s počtem normostran v kapitole.

Mezi textové údaje patří **počet znaků** `chapter.numberOfCharacters`, který byl spočítán při extrakci kapitol a **počet normostran** `chapter.numberOfStandardPages`, které jsou ze znaků spočítany pomocí vzorce 5.1.

Obrázky jsou do jednotlivých kapitol přiřazeny podle strany na které se nachází. Stránka na které se nachází se porovná se stránkovým rozsahem kapitoly a přičte se obsah obrázku do **celkového obsahu** obrázků a zvýší se **počet obrázků** `chapter.numberOfImages` v ka-

pitole. Počet normostran je pak přepočítán pomocí rovnice 5.6, kde $V$ je obsah obrázků kapitoly `chapter.imageVolume`.




Obrázek 5.7: Detail rozšířitelné sekce – Kapitoly. Každá kapitola zde má svůj blok obsahující informace o kapitole. Kapitola má dvě sekce: Text a Obrázky. Pod bloky kapitol se nachází celkový počet normostran z kapitol.

# Kapitola 6

# Testování

V rámci vývoje aplikace bylo provedeno testování za účelem zjištění přesnosti a výkonosti (kapitola 5.1) a uživatelské testování. Na základě uživatelského testování bylo provedeno několik změn v implementaci a také došlo ke změně virtuálního serveru.

## 6.1 Porovnání přesnosti s ostatními nástroji

Počet znaků dokumentu byl porovnáván s nástroji vyjmenovanými v kapitole 4.1. První z uvedených testování bylo provedeno na bakalářské práci[1]. Jak lze vidět v tabulce 6.1, *normostrana.cz* a *pocetznaku.cz* spočítaly stejný počet znaků, což je dáno tím že fungují na stejném principu – text se vkládá do textového pole. Po zkopírovaní textu bakalářské práce z dokumentu PDF bylo v některých případech vidět, že znak s diakritikou se rozdělil na dva znaky kvůli rozdílnému kódování. To je nejspíše jeden z důvodů proč je napočítáno o 414 znaků více než ve výsledném řešení. Další nástroj *word-count-character-count-tool* spočítal znaků výrazně více než ostatní nástroje, protože rozdělil znaky s diakritikou na dva znaky.

| **Webový nástroj** | *normostrana.cz* | *pocetznaku.cz* | *word-count-character-count-tool* | ***Výsledné řešení*** |
|---|---|---|---|---|
| **Počet znaků** | 44648 | 44648 | 51801 | 44234 |

Tabulka 6.1: Výsledky testu s bakalářskou prací. Výsledné řešení spočítá o 414 znaků méně než normostrana.cz a pocetznaku.cz, a výrazně méně než word-count-character-count-tool. Tyto rozdíly nejspíše způsobuje to, že v referenčních řeseních se některé znaky s diakritikou rozdělí na dva a také dochází k zalomení řádků.

V dalším testu byl použit dokument s přesně 1800 znaky, tedy jednou normostranou, z nichž 10 znaků obsahovalo diakritiku. Výsledky testu se nacházejí v tabulce 6.2. Nástroje *normostrana.cz* a *pocetznaku.cz* spočítaly o 12 znaků více, počítaly jako znak i prázdný řádek a opět došlo po zkopírovaní k rozdělení znaků s diakritikou. U *word-count-character-count-tool* byly započítány i pomlčky způsobené zalomením řádku a číslo stránky. Výsledné řešení započítalo navíc číslo stránky s mezerami okolo něj, tedy o tři znaky více, než v nahraném dokumentu.

[1] https://www.vutbr.cz/studenti/zav-prace/detail/78613

| **Webový nástroj** | *normostrana.cz* | *pocetznaku.cz* | *word-count-character-count-tool* | ***Výsledné řešení*** |
|---|---|---|---|---|
| **Počet znaků** | 1812 | 1812 | 1809 | 1803 |

Tabulka 6.2: Výsledky testu s jednou normostranou. Výsledné řešení je nejpřesnější, ale započítává i číslo stránky. Ostatní nástroje špatně počítají se zalomením textu nebo písmeny s diakritikou.

## 6.2 Uživatelské testování

Uživatelské testování proběhlo dvěma způsoby. Při prvním byl jeden uživatel pozorován při práci s nástrojem, druhý byl založen na otázkách pro uživatele – studenty vysokých škol dokončujících diplomovou práci. Zpětná vazba od uživatelů byla zohledněna i ve výsledném řešení.

### Průběh testování

Při pozorování byl vybrán uživatel, který dokončoval svou diplomovou práci. Dostal za úkol si průběžně kontrolovat počet normostran a byl při této činnosti pozorován. Na konci mu byly položeny stejné otázky jako respondentům kontaktovaným přes web. Otázky byly zaslány 5 studentům různých vysokých škol.

- Jak se vám v aplikaci orientovalo?
- Jaké nové funkce by jste přivítali?
- Jaký prvek nebo údaj se vám zdál nejužitečnější?
- Je nějaký prvek nebo údaj, který vám přijde zbytečný?
- Co se vám líbí na vzhledu stránky?
- Co by jste změnili na vzhledu stránky?

### Výsledky testování

Výsledky uživatelského testování pomohly doplnit funkcionalitu nástroje a změnit některé prvky, což odpovídá evaluační části vývoje jak ve své knize navrhuje Jan Řezáč. Několik uživatelů odpovídalo na otázky po přidání nových funkcionalit.

- Orientace:
  - Respondenti reagovali na úvodní stránku, která byla na základě reakcí přepracována. Na úvodní stránce by prý mohlo být více detailů o tom, jaké údaje nástroj poskytuje, zobrazených například ve formě snímků obrazovky. Dále by zde mohlo být uvedeno, z čeho všeho jsou výsledné normostrany počítány.
  - Původní nadpis stránky „Počítání normostran“ prý neodpovídá funkcionalitě, co tento nástroj nabízí. Byl proto změněn na „Analýza diplomových prací“.
  - Popisek „Celkový počet normostran z kapitol“ je blízko popisku „Celkový počet normostran“ a je to matoucí. Došlo ke zkrácení na „Počet normostran z kapitol“.
  - Při nahrávání by jeden uživatel navíc k loaderu přidal i popisek akce co právě probíhá, popisek ale nakonec nebyl přidán.

  - V aplikaci se dle uživatelů pro její jednoduchost snadno orientuje.
  - Při pozorování nebyl zjištěn žádný problém s orientací ve stránce.

- Funkce, prvky a údaje:
  - Pozorováním uživatele bylo zjištěno, že uživatel často sčítal počet normostran z kapitol, protože se jednalo o přesnější číslo, které nezapočítávalo úvodní text. To stejné potvrdilo i několik dalších respondentů, proto byla přidána funkcionalita pro počítání normostran z kapitol.
  - Pozorovaný uživatel se snažil omezit výskyt slov „pro“ a „bude“ pomocí grafu nejčastěji používaných slov. Tato funkce se líbila i respondentům, ale navíc by zde ocenili i přesný počet slov u každého sloupečku. Tento údaj byl přidán.
  - U ukazovátka poměru textu a obrázků by byl vhodný popisek s přesným číslem procent, který byl následně přidán. Několik uživatelů tuto funkcionalitu ocenilo jako nejlepší.
  - Grafy a digramy, které byly zakresleny vektorově, uživatelé vnímali jako obrázky, ale aplikace ne. Nástroj by také mohl poskytovat funkci pro ověření plagiátorství, tedy porovnání s ostatními pracemi. Tyto funkcionality nebyly přidány.

- Vzhled stránky:
  - Jeden z uživatelů by ocenil rozdělení tisíců od zbytku čísla. Toto vylepšení nebylo implementováno.
  - Uživatelům se líbila jednoduchost grafického vzhledu webových stránek, rozšiřovací sekce podle jejich názoru pomohly zpřehlednit statistiky.
  - Podle uživatelů byly ty nejdůležitější informace po nahrání souboru ihned viditelné.
  - Většina uživatelů by na vzhledu stránky nic neměnila. Jednomu uživateli přišlo rozhraní příliš „modré“, několik dalších naopak ocenilo použité barvy.

# Kapitola 7

# Závěr

Cílem této práce bylo vytvořit webový nástroj, který zpracuje soubor PDF a získá z něj různé charakteristiky a statistické údaje o rozsahu textu. Tento nástroj je užitečný hlavně proto, že při průzkumu nebyl nalezen žádný jiný nástroj, který by splňoval požadavky na výpočet normostran ze souboru PDF a ani nástroj umožňující přepočtení obrázků na normostrany.

Nejdříve jsem nastudovala způsoby návrhu webových aplikací a strukturu formátu PDF. Následně proběhl návrh aplikace a výběr technologií. Výsledný nástroj je schopen získat statistické údaje z textu, obrázků a kapitol první úrovně. Z textu se podařilo analyzovat údaje o znacích, slovech, větách a stránkách. Obrázky se povedlo extrahovat, převést na velikost na stránce A4 a s tím související statistiky, na základě získaných údajů se podařilo vytvořit přepočítávání obrázků na normostrany. Hlavním úspěchem bylo získání kapitol a statistik o kapitolách, které by ale ještě mohli být rozšířeny o podkapitoly. Nástroj byl průběžně vystavován pomocí cloudové platformy Heroku na adrese `http://standardpages.herokuapp.com/standardpages/`. Po implementaci proběhlo na této stránce uživatelské testovaní, které vedlo k některým změnám v uživatelském rozhraní a k přidaní nových funkcí.

Jedním z možných rozšíření této práce je možnost výběru i jiného formátu než je PDF a také přepočítání grafů, diagramů a algoritmů na normostrany. Aplikace by mohla mít možnost přihlášení. Po autentizaci by umožnila uživateli ukládat výsledky do vybrané nástěnky, kde by se nacházel přehled předchozích výsledků práce. Pro každou práci by si tak student mohl vytvořit nástěnku, která by zobrazovala pokrok a rozdíly mezi posledními nahranými verzemi a to jak v rámci celkového textu, tak v rámci kapitol. V nástěnce by měl možnost sdílet výsledky s ostatními uživateli – například student může sdílet nástěnku s učitelem, který tak má přehled o pokroku studenta. V rámci nástěnky by bylo možné nastavit hranice normostran, tedy minimální počet normostran, kterých má text dosáhnout a také maximum, které by neměl přesáhnout. Tyto údaje by poté sloužili k barevné signalizaci toho, jestli je počet normostran v požadovaném rozsahu.

# Literatura


[1] Django Software Foundation : The web framework for perfectionists with deadlines | Django. [Online; navštíveno 30.04.2019].
URL https://www.djangoproject.com/

[2] DjangoGirls: What is Django? [Online; navštíveno 30.04.2019].
URL https://tutorial.djangogirls.org/en/django/

[3] Hunter, J.; Dale, D.: The Matplotlib User's Guide. *Matplotlib 0.90. 0 user's guide*, 2007.

[4] Middleton, N.; Schneeman, R.: *Heroku: Up and Running: Effortless Application Deployment and Scaling*. "O'Reilly Media, Inc.", 2013, ISBN : 978-1-449-34139-8.

[5] Morris, S.: Tech 101: The Ultimate Guide to CSS. [Online; navštíveno 26.04.2019].
URL https://skillcrush.com/2012/04/03/css/

[6] Morris, S.: Tech 101: What is JavaScript? [Online; navštíveno 26.04.2019].
URL https://skillcrush.com/2012/04/05/javascript/

[7] Per, C.: HTTP Definition. [Online; navštíveno 28.04.2019].
URL https://techterms.com/definition/http

[8] Shannon, R.: What is HTML. [Online; navštíveno 26.4.2019].
URL http://www.yourhtmlsource.com/starthere/whatishtml.html

[9] Weinschenk, S. M.: *100 věcí, které by měl každý designér vědět o lidech*. Computer Press, 2011, ISBN 978-0-321-76753-0.

[10] Whitington, J.: *PDF Explained*. O'Reilly Media, Inc., 2011, ISBN 978-14-49321-58-1.

[11] Řezáč, J.: *Web ostrý jako břitva*. Baroque Partners, 2014, ISBN 978-80-87923-01-6.