České vysoké učení technické
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Nezalomitelné mezery v českém HTML
Non-breakable spaces in Czech HTML

Vypracoval

Jan Mucha

Bakalářská práce
odevzdávána při plnění požadavků bakalářského stupně studia.

Studijní program bakalářského stupně: Softwarové inženýrství a technologie

Praha, říjen 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Mucha** Jméno: **Jan** Osobní číslo: **503284**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Nezalomitelné mezery v českém HTML

Název bakalářské práce anglicky:

Non-breaking spaces in Czech HTML

Pokyny pro vypracování:

Typografická pravidla českého jazyka vyžadují přítomnost nezalomitelných mezer na specifických místech. Pro psaní HTML aktuálně nejsou k dispozici vhodné nástroje, které by tuto činnost automatizovaly (tak, jako např. program `vlna` pro TeX). Analyzujte problematiku automatického vkládání nezalomitelných mezer a implementujte ji, podle vlastní volby, jako jednu či více z následujících možností:
- rozšíření do textového editoru
- rozšíření do webového prohlížeče
- rozšíření do vývojářského IDE

Uvažte, která všechna pravidla je možné a/nebo praktické do algoritmu zahrnout. Vzniklý zdrojový kód publikujte na vhodném míst, v závislosti na povaze rozšíření.

Seznam doporučené literatury:

https://github.com/martykan/vlna-js/
https://prirucka.ujc.cas.cz/?id=880
http://ftp.linux.cz/pub/tex/local/cstug/olsak/vlna/
http://www.nedivse.cz/doplnovani-pevnych-mezer/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

RNDr. Ondřej Žára Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **28.08.2023** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **16.02.2025**

RNDr. Ondřej Žára		prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ Podpis studenta

Datum převzetí zadání

Vedoucí práce:
RNDr. Ondřej Žára
Katedra počítačové grafiky a interakce
Fakulta elektrotechnická
České vysoké učení technické v Praze
Technická 2
166 27 Praha
Česká republika

Zadání/Assignment

Nezalomitelné mezery v českém HTML

Typografická pravidla českého jazyka vyžadují přítomnost nezalomitelných mezer na specifických místech. Pro psaní HTML aktuálně nejsou k dispozici vhodné nástroje, které by vkládání těchto mezer automatizovaly (tak, jako např. program ‘vlna‘ pro TeX). Tato bakalářská práce analyzuje problematiku automatického vkládání nezalomitelných mezer a popisuje, která pravidla je možné a/nebo praktické do algoritmu zahrnout. Řešení je implementováno jako rozšíření do textového editoru Visual Studio Code. Vzniklý zdrojový kód je publikován na GitHubu.

Non-breaking spaces in Czech HTML

Typographic rules of Czech language require presence of non-breaking spaces at specific places. There is no generally available tooling suitable for automatic insertion of said spaces into HTML (similarly to what e.g. the ‘vlna‘ TeX program does). This Bachelor's thesis analyzes the concept of automatic non-breaking space insertion and describes which rules are feasible or suitable to be implemented. The resulting program is formed as a text editor Visual Studio Code extension and it's source code is published on GitHub.

Prohlášení

„Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.“

Praha, 1. září 2023

Poděkování

Děkuji všem svým blízkým za důležitou podporu. Též děkuji panu RNDr. Ondřeji Žárovi za skvělé vedení projektu.

Abstrakt a klíčová slova

Abstrakt (česká verze)

Tato práce se zabývá typografickými pravidly pro zalamování řádků v českém jazyce a jejich automatizovanou aplikací v dokumentech HTML. Výsledkem implementační části je rozšíření do textového editoru Visual Studio Code, které vyhledává místa, kde nemají být zalamovány řádky, a nahrazuje na nich klasické mezery těmi nezlomitelnými.
Klíčová slova – typografie, HTML, nezlomitelné mezery, zalamování textu, rozšíření, Visual Studio Code

Abstract (English version)

This thesis is concerned with finding places where non-breakable spaces should be inserted according to typographic rules for the Czech language in HTML documents. The goal of this thesis' implementation part is an extension of the text editor Visual Studio Code that inserts non-breakable spaces at these places.
Keywords – typography, HTML, non-breakable spaces, line wrapping, extension, Visual Studio Code

Contents

Kapitola 1

Úvod

Pro psaní entit NM v HTML je potřeba 5 znaků (středník není bezpodmínečně nutný; např. * (;)*), a psaní znaku pro NM nebo jeho hledání NM v mapě znaků jsou také zdlouhavé. Nástroj, který by to prováděl automaticky, může ušetřit nemalé množství času a energie uživatelů, a to i těm, kteří pravidla pro (ne)zalamování řádků dobře znají.

1.1 Motivace

Ve škole pro mě byla systematika českého jazyka jednou z nejzajímavějších oblastí. Motivace k této bakalářské prácitedy pramení z mé vášně pro český jazyk, ale také z mého smyslu pro pořádek a úpravu textů. Jelikož jsou webové stránky čteny nezřídka mnoha čtenáři, je politováníhodné, že jsou typografická pravidla někdy zbytečně opomíjena z důvodů nedbalosti či jejich neznalosti.

V rámcových vzdělávacích programech přitom zmínky o typografii začínají a končí praktickou stránkou úprav elektronických dokumentů. Pokud jde o pravidla typografie, RVP se o nich nezmiňují. [1, 2, 3, 4, 5]

Jelikož je text po tisíce let prostředkem nejrychlejšího předávání slovních informací[6] a jeho řádná úprava s sebou nese lepší čitelnost, dalším důvodem implementace této bakalářské práce je pro mě její přínosnost. Ačkoli se v krátkodobém horizontu může zdát slabá, v tom dlouhodobém se, předpokládám, čas a energie věnované této bakalářské práciv součtu vrátí uživatelům internetu.

1.2 Cíle

Hlavním cílem je ušetřit v současnosti stále rostoucímu množství lidí čas a energii při čtení textů na internetu i jejich psaní. Laťka pro splnění tohoto cíle je stanovena na 15 uživatelů produktu této bakalářské práceza prvních 60 dnů od jeho prvního vydání.

Druhým z cílů této bakalářské práce je zvýšit ve společnosti povědomí o typografických pravidlech českého jazyka. Především o existenci NM, které jsou součástí téměř každé

úpravy textu v elektronické podobě. Aby toho bylo dosaženo, musí být uživatelé zmíněného produktu pobízeni k přečtení typografických pravidel.

Part I

Teoretická příprava

Kapitola 2

Pravidla pro vkládání

V této kapitole jsou rozebírána pravidla českého jazyka pro vkládání NM. Ta jsou čerpána z technické normy ČSN 01 6910 pro úpravu dokumentů zpracovaných textovými procesory[1].

U jednotlivých pravidel je vyhodnocována náročnost a dopad jejich aplikace. Podle těchto měřítek jsou pravidla rozdělena do tří skupin. Je přitom bráno v úvahu, že chyby false negatives, *kdy NM není vložena tam, kde být má, nejsou svoji škodou tak závažné jako* false positives, *kdy NM je vložena tam, kde být nemá.*

2.1 ČSN 01 6910

České technické normy nejsou obecně závazné. ČSN 01 6910, ze které jsou čerpána pravidla, tedy obsahuje pouze množinu pravidel, která jen pomáhají uživatelům s úpravou dokumentů.[7]

Některá pravidla normy jsou zde, shodně jako v původním znění normy, uvedena jako doporučení, jiná jako požadavky. Důvodem je jejich různá důležitost a technické možnosti textových procesorů pravidla uplatnit.[7]

2.2 Seznam pravidel

Pro lepší přehled je zde vypsán seznam všech dohledaných typografických pravidel vztahujících se k problematice nezlomitelných mezer.

- Jistá pravidla
  - Členěná čísla
  - Měřítka a poměry
  - Rovinné úhly, stopy a palce

[1] Některá typografická pravidla, o nichž norma mluví a která se týkají zalamování řádků, zde nejsou popsána, jelikož na vkládání NM nemají žádný vliv.

  - Ampersand
  - Znaky hvězdička a křížek
  - Smíšená čísla
- Riziková pravidla
  - Čísla a zkratky
  - Počítané jevy
  - Zkratky rodných jmen a příjmení
  - Poštovní adresy
- Hrubá pravidla
  - Krátké předložky a slabičné spojky
  - Čísla a značky
  - Kalendářní data
  - Složené zkratky
  - Tituly před jmény
  - Vyjmenované následované zkratky
  - Matematické operátory
  - Pomlčky a znaky minus
  - Lomítka
  - Matematické závorky

2.3 Jistá pravidla

U pravidel této skupiny je možné spolehlivě zachytit většinu míst pro NM a zároveň nezpůsobovat žádnou škodu v podobě jejich vložení na nesprávné místo (false positives). Dodatečná konfigurace vkládání NM ze strany uživatele u těchto pravidel buď není potřeba, nebo je kvůli svému negativnímu dopadu na chybovost nevhodná.

2.3.1 Členěná čísla

Řádek se nemá zalomit v mezislovních mezerách, které jsou užity pro členění čísel, např. *7 200*, *12 345 678*, *26,144 537*. To platí i pro telefonní, faxová a jiná čísla členěná mezerou, např. *+420 776 665 443*, *906 059 066*, *900 44 55 44*.[8]

2.3.1.1 Vyhledávané výrazy

Pravidlo je aplikováno, kdykoli jsou v textu nalezena trojčíslí oddělená mezerou, nebo tvar telefonního či poštovního směrovacího čísla, ve kterých jsou takto dělena i dvojčíslí.

2.3.1.2 Komentář

Pokud je v textu překlep ve smyslu chybějící interpunkce mezi dvěma čísly, snadno může dojít k chybě *false positive*. Pravidlo je však zařazeno do skupiny jistých, protože překlepy mohou způsobovat problémy vždy a bylo by složité se jimi zabývat.

V případě takto popsaných regulárních výrazů mezera nebude vkládána mezi číslice dlouhého čísla, které není, ačkoli by mělo být, členěné mezerami. Program totiž NM vkládá jen namísto standardních znaků mezery.

2.3.2 Měřítka a poměry

Řádek se nezalamuje v mezislovních mezerách, které jsou užity mezi dvojtečkami a sousedními čísly při psaní měřítek a poměrů nebo při naznačení dělení, např. *mapa v měřítku 1 : 50 000*, *poměr hlasů 5 : 3*, *10 : 2 = 5*.[8, 9]

2.3.2.1 Vyhledávané výrazy

Jsou vyhledávána místa s číslicemi a znaky *:* oddělenými mezerami.

2.3.3 Rovinné úhly, stopy a palce

Při společném zápisu stupně, minuty a vteřiny rovinného úhlu musí stát na stejném řádku, např. *17° 15'*. Při společném zápisu stop a palců musí stopy i palce stát na stejném řádku, např. *12' 3"*. Místo uvedených znaků lze používat také značky jednotek *ft* a *in* nebo slovní vyjádření.[9]

2.3.3.1 Vyhledávané výrazy

Při společném zápisu se s výjimkou zápisu zeměpisných souřadnic oddělují mezerou stupně, minuty a vteřiny rovinného úhlu. ČSN 01 6901 nestanovuje způsob označení zeměpisných souřadnic ani počet míst, kterými se jednotlivé složky uvádějí.[9] Hledána tak budou místa s čísly následovanými znaky či výrazy pro stupně, minuty, vteřiny, stopy nebo palce.

2.3.4 Ampersand

Dojde-li ve výrazu s ampersandem k zalomení řádku, řádek se přednostně zalomí před ampersandem.[9]

2.3.4.1 Vyhledávané výrazy

Vyhledávání ampersandu a výrazu zakončeného číslem je triviální. Znaků pro ampersand je však více.[9]

2.3.5 Znaky hvězdička a křížek

Znaky hvězdička a křížek použité pro označení data narození a úmrtí se od následujícího údaje o datu narození nebo úmrtí oddělují mezerou. Znak a datum musí stát na stejném řádku.[9]

2.3.5.1 Vyhledávané výrazy

V textu bude potřeba hledat hvězdičky a křížky následované daty. Měsíc v datu může být psán číselně či slovně.

2.3.6 Smíšená čísla

Ve smíšených zlomcích se celá část od zlomku odděluje mezerou, je-li zlomek zapsán lomítkem a číslicemi, např. 2 1/2, 8 3/4. Celá část a zlomek musí stát na stejném řádku.[9]

2.3.6.1 Vyhledávané výrazy

Vyhledávat se bude číslo psané arabskými číslicemi následované mezerou, číslem psaným arabskými číslicemi, lomítkem, dalším číslem psaným arabskými číslicemi a nakonec mezerou či interpunkčním znaménkem.

2.3.6.2 Komentář

Podle ČSN 01 6910 se mezera nepíše, je-li zlomek zapsán pomocí zlomkové čáry, např. $2\frac{1}{2}$, $8\frac{3}{4}$.[9] Nicméně pokud by autor textu toto pravidlo neznal a mezeru zde psal, bylo by vhodné na takovém místě řádek nezalamovat a NM doplnit (nemám v úmyslu tyto mezery mazat, protože pak by bylo chování programu hůře pochopitelné pro uživatele a navíc složitější na implementaci).

Nebyly dohledány informace, které by vylučovaly možnost zápisu smíšených čísel se zlomkovou částí v jiném než základním tvaru. Smíšená čísla se složitou zlomkovou částí nebo částí obsahující neznámé při vyhledávání podle tohoto návrhu sice nalezena nebudou (vzniknou chyby *false negative*), avšak se nepředpokládá, že by se jednalo o častý jev.

2.4 Riziková pravidla

Pro některá z pravidel nelze bez obtíží implementovat takový algoritmus, na který by bylo možné se spolehnout a který by obyčejné mezery nahrazoval alespoň téměř všude, kde je to potřeba, a zároveň je nevkládal na velké množství míst, kde nejsou žádoucí. S ohledem na riziko *false positives* nelze ani nabídnout uživateli rozumnou konfiguraci, která by mu umožňovala zavést si byť jen částečnou aplikaci těchto pravidel.

2.4.1 Čísla a zkratky

Řádek se nemá zalomit v mezislovních mezerách, které jsou užity mezi číslem a zkratkou počítaného předmětu nebo písmennou značkou jednotek a měn, např. *5 str.*, *8 hod.*, *s. 53*, *č. 9*, *obr. 1*, *tab. 3*, *100 m²*, *10 kg*, *16 h*, *19 °C*, *1 000 000 Kč*, *250 €*.[8]

2.4.1.1 Komentář

Čísla psaná arabskými číslicemi mohou být v textu nacházena triviálně. Poněkud komplikovanější je však rozlišování římských čísel, značek a zkratek označujících jednotky či počítané subjekty. Zápis římských čísel navíc není jednotně standardizován a římské číslice je možné psát malými písmeny.[10] Složitým způsobem lze sice zachytit alespoň některá místa, kde má být toto pravidlo aplikováno, avšak bez analýzy kontextu ho nelze aplikovat spolehlivě bez *false positives*.

2.4.2 Počítané jevy

Řádek se nemá zalomit v mezislovních mezerách, které jsou užity mezi číslem a názvem počítaného jevu, např. *500 lidí, 365 dní, 10 kilogramů, strana 2, tabulka 3, 5. pluk, 8. kapitola, II. patro, Karel IV.*[8] Má-li však číslo alespoň 2 číslice a počítaný jev nebo jeho zkratka (bez případné tečky) alespoň 3 znaky, je dovoleno řádek zalomit.[9]

2.4.2.1 Komentář

Setkáváme se zde s podobnými problémy jako u pravidla čísel a zkratek, tedy že o výrazech vyskytujících se poblíž čísel nelze jednoznačně říci, zda s těmito čísly souvisí. Měn však mnoho není a lze je snadno implementovat pod pravidlem Čísla a značky.

2.4.3 Zkratky rodných jmen a příjmení

Řádek se nemá zalomit v mezislovních mezerách, které jsou užity mezi zkratkami rodných jmen a příjmeními, např. *J. Tříska, H. Strnadová*, oddělit příjmení od vypsaného jména se však připouští, *např. Jiří | Tříska, Helena | Strnadová.*[8]

2.4.3.1 Komentář

Příjmení, křestních jmen, a tedy i jejich zkratek, existuje prakticky neomezené množství. Na velká počáteční písmena také není možné spoléhat, jelikož zdaleka nemusí indikovat pouze jméno a příjmení.

2.4.4 Poštovní adresy

Stojí-li adresa na více řádcích, dílčí část adresy (např. ulice a číslo domu, název města a číslo městské části nebo PSČ a název adresní pošty) má být přednostně uvedena na

stejném řádku. Např. *Point, s. r. o., Václavské nám. 846/1, 110 00 Praha 1* nebo *Dr. Antonín Střelka. Kryštofovo Údolí 117, 460 01 Liberec.*

2.4.4.1 Komentář

Normy říkají, že pokud je adresa součástí souvislého textu, jednotlivé adresní údaje se oddělují čárkou.[9] Avšak i za předpokladu, že uživatelé budou správně oddělovat čárkami jednotlivé části, je spolehlivá detekce adres pomocí RV komplikovaná, ne-li nemožná, a RV tak nelze použít pro zajištění vkládání NM do výrazů mezi čárkami.

Ačkoli aplikaci tohoto pravidla technicky vzato lze konfigurovat, bylo umístěno do skupiny rizikových, protože se nepředpokládá, že by jakákoli jeho dodatečná konfigurace byla pro uživatele užitečná. Pro ideální vkládání NM na základě tohoto pravidla by uživatel musel každou adresu, která se v dokumentu vyskytuje, aplikaci sdělit, že konkrétní řetězec je adresa, a tak je v těchto případech mnohdy snazší doplnit NM ručně.

2.5 Hrubá pravidla

Při aplikaci poslední skupiny pravidel budou vznikat *false positives* nebo *false negatives*. Jelikož se u daných pravidel nepředpokládá, že se tak bude stávat příliš často, je možné uživateli nabídnout volbu, zda budou aplikována, a chybovost tak zmírnit.

2.5.1 Krátké předložky a slabičné spojky

Řádek se nemá zalomit v mezislovních mezerách, které jsou užity ve spojení neslabičných předložek *k*, *s*, *v*, *z*, slabičných předložek *o*, *u* a spojek *a*, *i* s následujícím slovem, např. *v cíli*, *s bratrem*, *k domu*, *z Prahy*, *u rybníka*, *o lidech*.[8]

2.5.1.1 Vyhledávané výrazy

Nalezení většiny dotyčných předložek a spojek, za kterými má následovat NM, je triviální. V případě předložek *v* a *i* je pak nutné rozlišovat římské číslice. Ty jsou obvykle, ačkoli ne vždy, psány velkými písmeny a obvykle stojí za počítanými jevy, tudíž ne na začátku věty.

Nalezené slovo $V/v/I/i$ mohou být považována za předložku či spojku, pokud platí:

- je následováno mezerou,
- tato následující mezera je následována slovem, případně arabskou číslicí
- a začíná (tedy je celé tvořeno) malým písmenem, nebo se nachází bezprostředně za tečkou a mezerou, což obvykle indikuje začátek věty, kde se římské číslice obvykle nevyskytují.

2.5.1.2 Komentář

V textu existuje riziko výskytu zápisu matematických a fyzikálních značek. NM pak budou vkládány i kolem značek proměnných/veličin zápisem odpovídajících tomuto pravidlu.

Zmíněné rozlišování krátkých spojek a předložek od římských čísel není dokonalé: Je-li předložka *v* nebo spojka *i* psána, např. v názvu, velkým písmenem a zároveň před ní není detekován začátek věty, algoritmus je odsouzen udělat chybu typu false negative. Navíc tečka následovaná mezerou nemusí nutně znamenat začátek věty, nýbrž např. zkratku tomuto římskému číslu předcházející. Tečka předcházející římským číslům *V* nebo *I* vede k chybám false positive.

Předložku *v* a spojku *i* tedy nelze snadno rozlišit od římských číslic. Proto bude vhodné uživatelům používajícím římské číslice nabídnout možnost deaktivovat vkládání NM za těmito slovy. Implicitně by vkládání mezer na tato místa mělo být zapnuté, jelikož očekávám, že výskyt římských čísel I a V, nebude příliš frekventovaný.

2.5.2 Čísla a značky

Řádek se nemá zalomit v mezislovních mezerách, které jsou užity mezi číslem a souvisící značkou, např. *10 %*, *§ 22*, *# 1*, ** 1890, † 1938.*[8, 9]

2.5.2.1 Vyhledávané výrazy

Program by mohl mít seznam značek často používaných vedle čísel a mohl by vědět, které jsou používány před a které za číslem. Kdyby program rozpoznal takovou značku, v závislosti na tom, v jakém směru by od značky bylo očekáváno číslo, podíval by se program po mezeře před nebo za touto značkou a čísle, jež předchází, či následuje. Jestliže by je našel, vložil by NM.

2.5.2.2 Komentář

Úskalí spočívá v množství značek, které se v textu potenciálně mohou nacházet. Značkám lze navíc někdy přisuzovat různé nové významy i měnit jejich pozici vzhledem k číslu.[11] Vznikají tak méně závažné chyby false negatives, ale také závažnější false positives.

Pokud však uživatel zavádí novou značku, nebo používá nějakou značku v jiném významu než který je obecně známý a uznávaný, může takovou chybovost předpokládat. Mohlo by být užitečné mu umožnit zapnout a později vypnout vkládání NM u značek podle jeho výběru.

2.5.3 Kalendářní data

2.5.3.1 Znění pravidla

Řádek se nesmí zalomit v mezislovních mezerách, které jsou užity v kalendářních datech mezi dnem a měsícem. Jestliže text není zarovnán do bloku, je při číselném zápisu vhodné

neoddělovat ani jeho ode dne a měsíce, např. 8. 5. | 1945, 17. listopadu | 1989.[8, 9]

2.5.3.2 Vyhledávané výrazy

Standardizovaných formátů, jakými lze data v češtině psát, je sice více, ale neměl by být problém zachytit je všechny. První až deváté dny i měsíce mohou být podle standardu psány dvoumístně. V případě číselně-slovního vyjádření se pro název měsíce používá slovo ve druhém pádu.[9] Pro účely této práce není nutné zabývat se zápisy bez mezer. Předpokládám, že pokud se uvede rok se 4 nebo více číslicemi, nebude členěn a budou uvedeny všechny jeho číslice, jak je zvykem a jak také přikazují ČSN.[9]

2.5.3.3 Komentář

Do regulárních výrazů pro datum bude někdy spadat také text, který nepředstavuje datum. Např. výčet čísel ve větách. Zároveň však v textu může být, ač to není obvyklé, úmyslně napsáno nějaké datum ironicky chybně. I tehdy se jedná o datum, a program by do něj tak měl NM vložit.

Uživatelům tedy může přijít vhod implicitně alespoň částečně validovat kalendářní data. Ignorovány budou sekvence jako např.
emph1. 13. a *32. 1.* Den by neměl být větší než 31 a v případě číselného vyjádření měsíce by toto číslo nemělo přesahovat 12. Na sekvence znaků, které nejsou vyhodnoceny jako validní data, při vkládání NM toto pravidlo nebude aplikováno. To může alespoň některým nesprávně vloženým NM v místech, kde se nejedná o datum, předejít.

Stejně tak bude vhodné nechat uživatele rozhodnout o vkládání NM pro připojení roku v číselných zápisech dat.

2.5.4 Složené zkratky

2.5.4.1 Znění pravidla

Řádek se nemá zalomit v mezislovních mezerách, které jsou užity ve složených zkratkách, v ustálených spojeních a v různých kódech, např. *a. s., s. r. o., j. č., n. l., T. G. M., J. F. K., PS PČR, FEL ČVUT, ČSN 01 6910*, složené zkratky a označení se v případě nutnosti doporučuje dělit podle dílčích celků, např. *ÚJČ | AV ČR, ČSN | EN | ISO 9001* (nikoli **ÚJČ AV | ČR, *ČSN EN ISO | 9001*),[8]

2.5.4.2 Komentář

Jelikož existuje nepřeberné množství ustálených slovních spojení a kódů, není pro mě reálné toto pravidlo v rozumné míře v bakalářské práci zpracovat. Je však možné nechat uživatele konfigurovat seznam regulárních výrazů z jeho textu spadajících pod toto pravidlo.

2.5.5 Tituly před jmény

Řádek se nemá zalomit v mezislovních mezerách, které jsou užity mezi zkratkou titulu nebo hodnosti uváděnou před osobním jménem, např. *p. Čečetková, mjr. Veselý, Ing. Poliaková*, lze však oddělit titul a rodné jméno od příjmení, např. *Ing. Ivana | Poliaková*.[8]

2.5.5.1 Komentář

Zkratka pro titul se svým zápisem může shodovat se zkratkou pro něco jiného a v tom se skrývá riziko *false positives*.

Zkratky titulů/hodností mohou být buďto následovány jménem či dalším titulem, nebo spojeny s dalším titulem pomocí ampersandu (&), či latinské spojky *et.*[9] Navíc tituly nejsou omezenou množinou – je třeba počítat se zaváděním nových včetně nekonvenčních. Nadto titul v závislosti na původu nemusí být vždy zapisován podle pravidel českého jazyka, nýbrž i jiného.

Chyb *false positive* však nepředpokládám mnoho, proto je ke zvážení nabídnout uživateli aplikaci tohoto pravidla jako volitelnou. Vyhledávány by byly jen nejběžnější tituly a hodnosti následované slovem s velkým počátečním písmenem, popř. hodnosti následované zkratkou jednoho z těchto titulů, v následujícím slově. Další tituly by uživatelé museli doplnit manuálně.

2.5.6 Vyjmenované následované zkratky

Řádek se nemá zalomit v mezislovních mezerách, které jsou užity mezi zkratkami čistě grafickými *tj.*, *tzv.*, *tzn.* a souvisícím výrazem, který za nimi bezprostředně následuje, např. *tzv. klikání*.[8, 11]

2.5.6.1 Vyhledávané výrazy

Vyhledávání několika běžných zkratek je triviální.

2.5.6.2 Komentář

Tyto tzv. zkratky čistě grafické lze vytvářet téměř z kteréhokoli slovního spojení. A pokud se takové uživatelem zavedené slovní spojení váže k následujícímu výrazu, vznikne chyba *false negative*. Při návrhu konfiguračních možností by to mělo být bráno v potaz.

2.5.7 Matematické operátory

Řádek může být v matematických výrazech zalamován kolem znaků $=$, $+$, -, $\pm$, v případě nutnosti je to dovoleno také kolem znaků $\times$, x, $\cdot$, $:$, $\div$ nebo $/$. V celém dokumentu má však být zalamováno jednotně před, nebo za těmito znaky. Zápisy rozměrů a tolerancí neobsahující slovní údaje (oproti např. *230 V s relativní tolerancí ±5 %*) mají stát na

stejném řádku. Jestliže je z důvodu zarovnání odstavce nezbytné, aby byl výraz rozdělen do dvou řádků, řádek se zalomí před znakem krát nebo znakem plus minus.[9]

2.5.7.1 Vyhledávané výrazy

Vyhledávání znaků $=$, $<$, $\leq$, $>$, $\geq$ a matematických operátorů s výjimkou znaků minus a lomítko je triviální. Problematika znaků minus a lomítko je rozebírána v částech Pomlčky a znaky minus a Lomítka. NM k nim pod tímto pravidlem vkládány nebudou.

Rozlišovat zápisy rozměrů a tolerancí od ostatních matematických zápisů by bylo komplikované, a tak to v této bakalářské práci není řešeno.

V dokumentu je zalamováno buďto před matematickými operátory a NM jsou vkládány za ně, nebo je zalamováno za nimi a NM jsou vkládány před ně. Ani ve druhém případě není NM vložená před matematický operátor v zápise rozměrů nebo tolerancí chybou (*false positive*), jelikož tyto zápisy by neměly být zalamovány vůbec, pokud to není nezbytné. Nicméně je předpokládán výskyt *false negatives*, jelikož zápis rozměrů a tolerancí je obtížné rozlišovat, a bude v nich tak NM kolem matematického znaménka vkládána jen z jedné strany.

Nezbytnost rozdělení výrazu do dvou řádků z důvodu zarovnání odstavce je chápána jako přesah délky zápisu nad délkou řádku, je považována za výjimečnou situaci a není brána v potaz.

Nabízí se umožnit uživateli vybrat si vkládání NM buďto před, nebo za matematickými operátory.

2.5.8 Pomlčky a znaky minus

Řádek se nemá zalomit v mezislovních mezerách, které jsou užity před pomlčkou, jež je od okolního textu oddělena z obou stran mezerami.[8]

2.5.8.1 Vyhledávané výrazy

Hledána je sekvence znaků mezera - pomlčka - mezera. NM nahrazuje jen první mezeru.

2.5.8.2 Komentář

Pomlčka je autory textů někdy psána jako podobně vypadající znak spojovník a minus (jde o jeden znak používaný jako spojovník i minus). Nicméně spojovník je psán bez mezer, tudíž, pokud je obklopen mezerami, lze ho považovat za nesprávně psanou pomlčku nebo minus. Ty je od sebe potřeba odlišovat jen někdy.

Jelikož má být NM vkládána před pomlčku, v případě, že se v dokumentu zalamuje za matematickými operátory, není z hlediska vkládání NM potřeba rozlišovat, zda jde o pomlčku, či minus, a NM lze spolehlivě vložit před tento znak.

Uživatel ale může zalamovat před matematickými operátory a technickými normami je (v textech administrativní povahy) dovoleno nahrazovat znak minus znakem pro pomlčku.[9] Význam znaku lze tak jen odhadovat – např. podle toho, zda napravo od něj

stojí číslo. Tento odhad dává podnět k chybám *false positives* i *false negatives*. Bylo by tedy vhodné uživatelům poskytnout možnost NM kolem znaků pro pomlčku, spojovník a minus nevkládat pro případ, že v dokumentu je zalamováno před matematickými operátory.

2.5.9 Lomítka

Doporučuje se řádek nezalamovat před lomítkem s mezerami dělícím dva výrazy.[9]

2.5.9.1 Vyhledávané výrazy

Hledána je sekvence znaků mezera – jedno či dvě lomítka – mezera. NM nahrazuje jen první z mezer.

2.5.9.2 Komentář

Pravidlo se netýká jen lomítek používaných jako matematické operátory dělení. Lomítko a dvě lomítka, obklopená mezerami, mají kromě matematických operací i další druhy použití: naznačení alternativy, grafické oddělení výrazů, naznačení veršování/řádkování – při úsporném psaní se v citaci poezie jedním lomítkem oddělují verše a dvěma sloky, v běžném textu se jimi mohou naznačit zalomení řádků, popř. odstavců.[9]

Jelikož má být NM vkládána před lomítko, v případě, že se v dokumentu zalamuje za matematickými operátory, není z hlediska vkládání NM potřeba rozlišovat, zda jde o matematickou operaci, či nikoli, a NM lze spolehlivě vložit před tento znak.

Uživatel ale může zalamovat před nimi. Ačkoli normy nemluví jednoznačně, u dvou lomítek vedle sebe se předpokládá, že do tohoto pravidla spadají, neboť jde o graficky podobnou značku jako jedno lomítko. Naproti tomu význam jednoho znaku lomítka s mezerami lze (stejně jako u pravidla Pomlčky a znaky minus) jen odhadovat např. podle toho, zda napravo od něj stojí číslo. Tento odhad by přinášel riziko *false positives*, a pokud se v dokumentu lomítko užívá pro matematické dělení, i *false negatives*. Bylo by tedy vhodné nechat uživatele vybrat, zda NM kolem lomítek nevkládat, pokud v dokumentu není zalamováno za matematickými výrazy, nýbrž před nimi.

2.5.10 Matematické závorky

Řádek se nemá zalomit uvnitř matematického výrazu v závorkách, není-li to nezbytné.[9]

2.5.10.1 Vyhledávané výrazy

NM budou vkládány namísto veškerých mezer v závorkách obsahujících alespoň jeden znak $=$, $+$, -, $\pm$, $\times$, x, $\cdot$, $:$, $\div$ nebo $/$, a dále jen mezery, číslice nebo běžná písmena pro označování proměnných: a, b, c, k, l, m, n, x, y, z.

2.5.10.2 Komentář

Jelikož se regulární výrazy mohou být někdy velmi komplikované, výše je popsán regulární výraz jen pro hrubé odfiltrování. Ačkoli se předpokládá, že by mohl existovat nějaký regulární výraz s přesnějšími výsledky, jelikož by to bylo složité, je vhodné alespoň umožnit uživatelům aplikaci tohoto pravidla vypnout.

2.6 Závěr

Vzhledem k povaze pravidel jsou RV užitečným prostředkem pro vyhledávání míst v textu, kam mají být vloženy NM. Některá pravidla nelze zachytit pomocí implicitních RV a pro jejich optimální aplikaci je potřeba uživatelská konfigurace, kterou více rozebírá kapitola 8, na str. 37).

Kapitola 3

Existující řešení

V odstavcích této kapitoly jsou vyhodnocovány silné a slabé stránky způsobů a nástrojů pro vkládání NM do HTML dokumentů, které jsou v současnosti dostupné.

3.1 Manuální a pseudo-manuální řešení

Ručně vkládat mezery během psaní programu může autora zdržovat. Urychlení si lze představit v aplikacích pro nastavení klávesových zkratek nebo příkazových maker. Ačkoli s nimi lze NM vkládat ručně rychleji než vypisováním jednotlivých znaků entit pro NM (více o zápisech NM viz kapitola 6 na str. 27), stále se nejedná o nic pohodlného, zvláště pokud autor není zvyklý se psaním NM zabývat.

3.2   replacer

Tato webová aplikace slouží pro automatické vkládání mezer do textu. Odpovídá tak do jisté míry zadání mé bakalářské práce. Po stisknutí tlačítka jsou v zadaném textu vyhledány sekvence definované v jedním z několika slovníků z nabídky. V takových sekvencích jsou pak vkládány NM a výsledný text.[12, 13] Aplikace se prezentuje jako určená pro práci s HTML a XHTML.[14]

3.2.1 Pozorované chování

Poté, co byla aplikace podrobena krátkému průzkumu, ukázalo se následující:

1. Program s textem pracuje pouze jako s prostým textem a značky HTML nijak nevyhodnocuje. Vznikají jak *false negatives* – značky mohou přerušit sekvenci, kterou program podle slovníku vyhledává – tak také *false positives* - např. atributy značky mohou obsahovat sekvenci, již nástroj považuje za předložku, a NM je tam tak nesprávně vložena.

2. Program nenahrazuje nezlomitelnou mezeru za standardní v místech, kde standardní stačí.

3. Lze si zvolit jeden ze 3 slovníků: *všude*, *typograficky* a *úzký sloupec* (seřazeny sestupně podle počtu obsažených slov). Úzký sloupec obsahuje 8 předložek a spojek, největší celkem přibližně 160 slov (včetně variant s velkými počátečními písmeny).[14])

3.2.2 Aktualizace

Aplikace vznikla v roce 2008 a dodnes byla každým rokem provedena alespoň drobná úprava týkající se většinou uživatelského rozhraní.[12] Autor poskytuje svůj kontakt a odkaz na formulář pro sdílení případných komentářů, názorů a návrhů na úpravu[13] a podle poznámek k verzím na přání uživatelů reaguje.[12] Tím se však často odchyluje od pravopisných a typografických pravidel.

3.3 Editory s funkcí najít a nahradit

Jedním z řešení je využití funkce pro nahrazování textu, kterou nabízí např. v aplikaci Word, a to v desktopové i webové variantě. Aplikace Notepad++ a online aplikace Overleaf, v níž je psána tato bakalářská práce, dokonce umožňují vyhledávat a následně nahrazovat části textu pomocí regulárních výrazů. Funkce *najít a nahradit* nepracuje se značkami a strukturou dokumentů HTML, ale může být postačující.

Další nevýhodou může být doba strávená otevíráním těchto nástrojů, v případě delšího textu může tuto dobu prodlužovat formátování a vykreslování textu. Na mém počítači je na tom s rychlostí otevírání nejlépe Notepad++, v němž se (při implicitním nastavení) klávesovou zkratkou ctrl + N otevře nový prázdný soubor prakticky okamžitě a zkratkou ctrl + H je možné otevřít okno funkce.

3.4 Vlna

Vkládání nezlomitelných mezer, avšak do zdrojových souborů TEXu, se věnuje program Vlna. Verze 1.5 od RNDr. Petra Olšáka přidává vlnky místo původních mezer za neslabičné předložky. Program je ovládán prostřednictvím příkazové řádky. Jeho online varianta je dostupná také online.[15] [1]

Program Vlna je v oblasti české typografie a sazby textu zřejmě nejznámější (je zmiňován na internetových fórech) a z nalezených nástrojů pro vkládání NM do českého textu funkcionalitou nejpokročilejší. I proto se na tento program na některých místech ve své bakalářské práci odkazuji.

[1] Vlna je dostupná na adrese https://martykan.github.io/vlna-js/.

3.4.1 Deklarovaná funkcionalita

Nástroj Vlna je schopný nahrazovat pouze ty mezery, jež se nacházejí bezprostředně za vybranými jednopísmennými slovy. Implicitní sada spojek a předložek, za nimiž jsou mezery nahrazovány, je tvořena slovy *K*, *k*, *S*, *s*, *V*, *v*, *Z*, *z*, *O*, *o*, *U*, *u*, *A* a *I*. Ostatní gramatická pravidla pro nezalamování řádků zřejmě nebyla považována za důležitá a při vkládání NM nebyla zohledněna.

Vlna nevkládá NM do bloků označených jako matematické výrazy nebo do bloků verbatim režimu (\verb<zn>...<zn>, \begtt...\endtt).

Program lze přepínačem přizpůsobit pro vkládání NM do dokumentů psaných v LaTeXu. Při přepnutí do LaTeX režimu program nenahrazuje mezery ani v dalších úsecích LaTeXu označených příkazy \begin a\end jako *display*, *equation* nebo *verbatim*.

Program lze také přepnout do webového režimu: K blokům, ve kterých NM nejsou vkládány, je v něm přidáno další označení verbatim módu. V tomto módu se ve webových souborech neobjeví kód s vlnkami, nýbrž jen komentáře.[16]

3.4.2 Pozorované chování

Jelikož je Vlna psána v jazyce CWEB, který mi není známý, zdrojový kód nebyl zkoumán. [17] Avšak na základě krátkého pozorování chování byly učiněny tyto praktické závěry:

1. Pokud je hledaná předložka od mezery oddělena znaky příkazu pro formátování písma, není tato mezera nahrazována vlnkou.

2. Program nenahrazuje nezlomitelnou mezeru za standardní v místech, kde standardní stačí.

3. Za předložky jsou považovány i římské číslice *I* a *V*, pokud se bezprostředně před a bezprostředně za nimi nachází mezera.

4. NM jsou sice vkládány za velkou spojku *A*, nikoli však, je-li psána malým písmenem.[16]

5. Speciální znaky používané pro značky přerušují sekvence znaků, které by jinak spadaly pod dané pravidlo pro vkládání NM. Vlna tedy nepracuje se sjednocenými texty částí dělených speciálními znaky, nýbrž s textem jako s jedním celkem a značky v textu nepřeskakuje, tudíž kolem nich NM nevkládá.

3.4.3 Konfigurovatelnost

Pomocí různých přepínačů lze například změnit množinu jednopísmenných předložek, za nimiž mají být NM vloženy, nastavit vstup a výstup (Vlna umí pracovat se soubory a konzolí), nebo dokonce nastavit libovolnou sekvenci znaků, kterou jsou mezery nahrazovány (zapisovanou v kódování daného souboru, např. v případě UTF-8 by pro entitní zápis NM v HTML byl použit příkaz „vlna -x 266E6273703B“). [16]

3.5 Seznam Médium

Články internetových médií obsahují NM příznačně často. Otázku, zda je dovede redakční prostředí vkládat automaticky, nebo je píší autoři ve snaze napsat co nejupravenější text, však nedokáže zodpovědět pouhá studie textu samotných publikací.

Po registraci na Seznam Médium bylo možné výše uvedené schopnosti tohoto prostředí testovat. Při psaní článků jsou k dispozici 4 textová pole: titulek, popisek hlavního obrázku, úvod článku, obsah článku. Všechna čtyři jsou při ukládání článku procházena pro automatické vložení NM. [18]

3.5.1 Pozorované chování

Toto automatické vkládání NM nelze nijak konfigurovat. Podle mého krátkého pozorování jsou NM vkládány za písmena *K*, *k*, *S*, *s*, *V*, *v*, *Z*, *z*, *O*, *o*, *U*, *u*, *A*, *a*, *I*, *i* oddělená od okolního textu mezerami, a to standardními mezerami, což je někdy podstatné ve chvíli, kdy např. několik těchto předložek a spojek následuje po sobě: „A v — takovéto větě pak vznikne chyba,“ – za druhým z výrazů je zde ponechávána standardní, zlomitelná, mezera.

Dále jsou NM vkládány do mezer členících čísla a také za tato čísla, třebaže se za nimi nenachází počítané jevy. Např. u věty „V kapitole 10 se nachází srozumitelný výklad,“ algoritmus vkládá NM mezi číslo „10“ a zvratné zájmeno „se“, avšak správně má být vložena mezi slova „kapitole“ a „10“.

Ze mně neznámých příčin však nejsou NM vkládány za zkratky *tzv.*, *tzn.*, *tj.*, *např.* Podobně nástroj neřeší složené zkratky jako *AV ČR* a nechává je rozdělené klasickou mezerou.

Uživatel může NM vkládat také manuálně, avšak tyto mezery nejsou prostředím nijak zvýrazněny.[18]

3.6 Závěr

Zkoumané nástroje zanedbávají určitá pravidla typografie, a do některých míst, kam NM náleží, je tak nevkládají. Funkce *najít a nahradit* je nedokonalá, jelikož je uživatel nucen zadávat různé sekvence podle typografických pravidel. S výjimkou Seznam média není efektivita práce těmito způsoby ideální.

Nástroj vytvořený v rámci této bakalářské práce by tedy měl dosavadní dostupná řešení překonávat svoji funkcionalitou a čas uživatele potřebný pro vložení NM by v ideálním případě měl být tak krátký jako v případě Seznam média. Pro vložení NM by tedy mělo stačit dokument uložit, použít klávesovou zkratku či napsat příkaz přímo v okně editoru.

Kapitola 4

Rozpoznávání českého textu

Ačkoli by se mohlo zdát, že vzhledem k povaze aplikace není potřeba kontrolovat jazyk dokumentu (autor se sám rozhoduje, kdy rozšíření použít), je vhodné uživatele na používaný jazyk upozornit v případě, že rozšíření nepodporuje jazyk do dokumentu, který není v češtině. V této kapitole se nachází způsoby jak rozeznávat češtinu, jakožto jediný podporovaný jazyk, uvnitř HTML dokumentu.

4.1 Potřeba filtrování HTML značek

Prostý text je v HTML dokumentech rozšířen o strukturu definovanou HTML značkami. Je nežádoucí vkládat do nich nezlomitelné mezery, a proto má smysl uvažovat o jejich filtrování od prostého textu.

4.1.1 Ignorování HTML značek

Nejjednodušším přístupem je však aplikovat pravidla vždy na celý dokument včetně textu značek HTML. Místa pro vkládání NM budou hledána pomocí vzorů a regulárních výrazů, které se v HTML příliš často nevyskytují. Pokud by ale taková situace nastala, mohou se do dokumentu zanést závažné chyby, které naruší jeho integritu. Značky s NM nepracují a nelze se spoléhat, že budou editační nástroje při jejich vložení kontrolovat, zda je NM uvnitř značky. Ano, na chyby dokáží upozornit validátory[19] a díky entitnímu zápisu NM v HTML lze tyto chyby relativně snadno najít. Nicméně uživatele by něco tak neočekávaného mohlo zklamat.

4.1.2 Odfiltrování HTML značek

Značka je zapisována pomocí znaků < a >. Může být buď párová, nebo nepárová. Při odfiltrování značek je předmětem vkládání NM veškerý text mezi značkami <body> a </body>, který není součástí značek HTML. Ty je při vkládání NM potřeba ignorovat, ale to není složité.

4.2 Český jazyk a cizí jazyk

Existuje více způsobů jak český jazyk rozpoznávat, které jsou zde popsány. Zajímavý je přitom tento pohled na chybovost: Chyba *false positive*, kdy se rozhodneme vkládat mezery v cizojazyčném textu, není závažná. Nejpoužívanějším cizím jazykem na internetu je totiž angličtina, a jelikož místa pro vkládání NM hledáme podle vzorů a regulárních výrazů, je v takovém textu velice vzácné vložení na místo, kde NM být nemá. Takové vzory jsou totiž v anglickém textu často podobné, nebo se zde obvykle nevyskytují.[20] Totéž se předpokládá i u ostatních jazyků. Je tedy možné, že se ve zpracovaném cizojazyčném textu občas vyskytne chyba, avšak nepředpokládám, že takových míst bude mnoho. Horší dopad by měla chyba typu *false negative*, kdy NM nejsou vkládány do českého textu.

4.2.1 Atribut lang

Pro účely rozpoznávání jazyka lze využít značky HTML. Jedním z prostředků, kterými lze jazyk poměrně spolehlivě určit, je konkrétně atribut lang, který je používán pro definici jazyka, v němž je psán text uvnitř značky. Může být použit jako atribut každé párové značky, ale u značky <html> je povinný. Atributy lang potomků přepisují atributy lang svých předků.[21] Průzkum ukázal, že všech 10 nejnavštěvovanějších webů České republiky a 8 z 10 nejnavštěvovanějších webů na světě používá (ve značce *<html ...>*) tento atribut samotný, nebo ve spojení s atributem xml:lang stejné hodnoty. Zbylé dva nejnavštěvovanější světové weby neobsahují latinku, tudíž prakticky nejsou předmětem vkládání NM.[22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39]

4.2.2 Atribut xml:lang

Atribut xml:lang dělá sice totéž co atribut lang a při psaní v XHTML má atribut xml:lang přednost před atributy lang v HTML, nutno však dodat, že xml:lang je označen jako zastaralý a používán je z důvodů kompatibility.[40] Vzhledem k tomu riziko, že text bude mít nastaven HTML atribut lang na *unknown* (kvůli validitě) a jazyk bude definován jen pomocí xml:lang, není velké, obzvláště pokud program budou používat autoři nově vznikajících stránek, a s časem se předpokládá jeho pokles.

4.2.3 Hlavička Content-Language

V případě, že je dostupná hlavička odpovědi serveru, indikátory jazyka dokumentu lze hledat také v ní. Hlavička Content-Language obsahuje informaci o jazyku čtenářů, pro které je dokument určen, tzn. neodpovídá přímo na otázku, ve kterém jazyce je text v dokumentu napsán, ale přesto je tato informace vypovídající. Nicméně tato hlavička nemusí být v odpovědi obsažena, (což značí obsah určený pro veškeré publikum.)[41]

Vzhledem k povaze aplikace – jde o nástroj pro autory HTML dokumentů (více viz kapitola 7 na str. 29) – však tuto cestu nelze použít.

4.2.4 Statistická analýza textu

Dalším způsobem, jak určovat jazyk, je pozorování a vyhodnocování statistických údajů. Každý jazyk má svá statistická specifika.

Mezi ta snáze aplikovatelná patří výskyt speciálních znaků. V češtině se diakritická znaménka vyskytují velmi často. Bohužel se na ně však nelze vždy plně spolehnout, jelikož ne každý je používá, a tak je ne každý český text obsahuje.

Jinými statistickými hodnotami, jež lze sledovat, jsou frekvence znaků, n-gramů a slov. Nicméně za určováním jazyka pomocí nich může stát složitější logika a jejich spolehlivost závisí na množství vyhodnocovaného textu. Pokud ho stránka mnoho neobsahuje, sice NM vlastně ztrácí na významu a nemusely by být vkládány. Text uvnitř každé HTML značky však teoreticky může být psán jiným jazykem, a tak nelze určit vše spolehlivě ani na stránce s delšími texty, jsou-li rozděleny na krátké texty pod různé značky. Pokud by byl vyhodnocován vždy všechen text stránky naráz, statistických dat by byl dostatek na drtivé většině stránek, kde dává úprava zalamování smysl vzhledem k množství textu, kvalita těchto dat by tím však klesla.

4.3 Rozhodnutí o implementaci

Bude implementováno vyhledávání značek <*body*> a </*body*> označujících tělo HTML dokumentu a využito jejich atributu *lang*, jelikož je to spolehlivější a snazší než metody založené na statistice. Na jazyk uvedený v potomcích značky těla dokumentu přitom nebude brán zřetel, jelikož u nalezených stránek, kde mají tyto značky jinou hodnotu atributu *lang*, slouží např. pro orientaci cizojazyčných návštěvníků a změnu jazyka stránky,[42, 43] a tudíž obsahují jen zanedbatelné množství textu.

Program Vlna v TEXu podle mých pozorování (viz kapitola 3 na str. 19) pouze vyhledává definované sekvence znaků mezera – předložka či spojka – mezera. U některých bloků označených jistou sekvencí znaků však vkládání mezer vypíná. Cílem této bakalářské práce není vytvořit v tomto ohledu lepší aplikaci než program Vlna, ale vzhledem k zadání (str. iii) se považuje za samozřejmé NM nevkládat do textu tvořícího značky – mezi znaky < a >. Ve validním dokumentu by to nemělo být složité. Jeho validitu však bude třeba před vkládáním NM nejprve zkontrolovat.

Kapitola 5

Práce s členěným textem

Text v HTML dokumentu může být místy přerušován značkami, které mohou obsahovat další text a jejichž nejznámějším příkladem je asi .... Tato značka může označovat např. slovo, které má být jinak formátováno.[44] V této kapitole je rozebírán princip práce s úryvky textu v jednotlivých značkách HTML a vkládání NM do míst v okolí těchto značek.

5.1 Vkládání NM do veškerého textu

V ideálním případě budou pravidla pro zalamování řádků správně aplikovaná i na rozhraní značek. jednou možností je pracovat s veškerým textem dokumentu globálně, nikoli po částech. K tomu je nutné pospojovat všechny části tohoto textu v jeden celek.

Popisovanou funkcionalitu pospojování přitom může usnadnit patřičné rozhraní pro práci s DOM, pokud bude takové k dispozici. V opačném případě lze postupovat následovně: Na začátku je veškerý text v těle dokumentu. Algoritmus si zapamatuje místa, kde se v textu jednotlivé značky nachází, a jejich znění. Následně jsou z textu vyjmuty, čímž je dosaženo celistvého textu bez značek. Jsou vloženy NM a nakonec jsou značky doplněny zpět. Jelikož je zbytečné pracovat s počty znaků tvořících značky, pořadí tohoto doplňování závisí na pořadí, v jakém byly zapamatovány a mazány.

Tento vztah podrobněji popisuje tabulka 5.1. Sloupec Alternace říká, zda je zapamatování značek provedeno spolu s mazáním postupně u každé značky (s alternací), nebo jsou nejprve všechny značky zapamatovány a následně všechny smazány (bez alternace). Varianty pořadí, které v tabulce nejsou uvedeny, vyžadují nadbytečnou implementaci.

5.1.1 Místa s odřádkováním způsobeným značkami

Pokud se ignorováním značek vloží NM na místo, za nebo před kterým je prostřednictvím značek implikováno odřádkování, není to na závadu, jelikož takové odřádkování se v souvislém textu neprojeví jako mezera, u které by bylo riziko, že bude nahrazena NM.

Pořadí zapamatování	Pořadí mazání	Alternace	Pořadí vkládání
Dopředné	Dopředné	Ano	Zpětné
Zpětné	Zpětné	Ano	Dopředné
Dopředné či zpětné	Zpětné	Ne	Dopředné

Table 5.1: Závislost mazání a doplňování značek

5.2 Vkládání NM do jednotlivých uzlů

Pro práci s jednotlivými uzly DOM je často k dispozici užitečné API.[15] Jednou možností je takové využít a zpracovávat text každého uzlu DOM zvlášť. Pak se však stává vkládání NM okolo značek v textu nedosažitelným, tzn. NM obvykle nebudou vloženy mezi slova umístěná v různých značkách, třebaže tam patří.

Podle mého pozorování to odpovídá např. také chování programu Vlna. (Více viz kapitola 3, str. 18.)

5.3 Rozhodnutí o práci s členěným textem

Složitost práce s členěným textem závisí na API, které bude k dispozici. Vkládání do každého uzlu zvlášť však nemá výrazný dopad na kvalitu, jelikož jde hlavně o text uvnitř odstavců a předpokládá se, že tento text je značkami členěn jen málokdy.

Kapitola 6

Mezery

Jelikož existuje několik druhů mezer a možností zápisu těch nezlomitelných je v HTML více, jsou popsány v této kapitole.

6.1 Vlastnosti speciálních mezer

Většina druhů mezer definuje svoji šířku relativně vzhledem k šířce písmene *m*. Jedná se mj. o em mezery, en mezery, mezery třetinové, čtvrtinové, šestinové, tenké a jemné. Číslicové mezery by měly mít šířku shodnou s šířkou číslic.[45] Pevné mezery jsou podobné těm nezlomitelným, ale od standardních mezer se liší navíc tím, že nemění svoji šířku při změně šířky řádku, pokud je text zarovnán do bloku. Druhů mezer je však ještě více.

6.1.1 Nezlomitelné mezery

Nezlomitelné mezery pro čtenáře vypadají na první pohled stejně jako běžné mezery. Mají ale tu vlastnost, že se na jejich místě nezalamuje řádek. Webové prohlížeče (konkrétně byla provedena zkouška na Google Chrome a MS Edge) při hledání míst, kde mohou zalomit řádek, tyto mezery ignorují. V případě, že se řádek na stránku nevejde a prohlížeč nenašel místo vhodné pro jeho zalomení, řádek nechá pokračovat nezalomený mimo viditelnou část stránky.

6.2 Zápis

Speciální znaky v HTML lze zapsat pomocí entit. Ty jsou definovány jako řetězce znaků začínající ampersandem (*&*) a středníkem (*;*).[46] Lze používat také entity na bázi ASCII a hexadecimálního zápisu ASCII. Některé speciální znaky lze navíc zapsat entitami zkratkové povahy.

Například NM lze v HTML zapsat následujícími způsoby: * * * * * * a * *.[15, 47] Z důvodu zpětné kompatibility může být v těchto zápisech vynechán

středník.[48] Nicméně za takovým zápisem podle výsledků experimentu v prohlížeči Google Chrome nesmí v případě ASCII entity bezprostředně následovat znak číslice (0-9, A-F). Další způsob zápisu speciálních znaků, především mezer, může určovat používaný editor. Mezi entitami HTML není rozlišován znak pevné mezery a NM, ačkoli ve skutečnosti by se měly lišit šířkou.

6.3 Rozhodnutí o zacházení se speciálními mezerami

Pracovat se všemi různými typy mezer by bylo velmi obtížné – bylo by nutné navrhnout, popsat a implementovat způsob, jak aplikovat pravidla pro každý jednotlivý typ mezery. Program Vlna nenabízí žádný způsob, jak se všemi speciálními mezerami pracovat[16] a od této bakalářské práce není vyžadován v tomto ohledu pokročilejší program. Cílem je aplikace, která dobře zvládá jednu věc, a tou jsou nezlomitelné mezery, které se navíc od těch pevných v HTML neodlišují.

Uživatelům může být užitečná možnost zapnout režim, ve kterém by program nahrazoval také speciální mezery a entitami psané standardní mezery, dostupná v nastavení.

Implicitní entitou pro vkládání NM bude * *. Pro nové uživatele je to asi nejpřijatelnější varianta, jelikož její zápis napovídá, že jde o *non-breaking space*, tedy NM. Uživatel by však v případě potřeby měl mít možnost zvolit, jakými znaky budou mezery nahrazovány, podobně, jako to umožňuje např. program Vlna.[16]

Kapitola 7

Typ aplikace

Tato kapitola je věnována uživatelským prostředím, pro která může být program mé bakalářské práce používán, a porovnávání jejich výhod a nevýhod.

7.1 Hodnoticí kritéria

Hlavním cílem je prakticky co nejvíce zvýšit počet zobrazení HTML dokumentů s vloženými NM. Od tohoto cíle jsou odvozena následující hodnotící kritéria, která na jeho splnění mohou mít vliv.

Potenciální uživatele přitom dělím na dvě skupiny. Čtenářem je člověk, který čte stránky a vložení NM mu má přinést pohodlí při čtení. Autorem je ten, kdo vytváří text nebo HTML dokument. V případě, že je uživatelem autor, vložené NM poslouží čtenářům jeho stránky.

7.1.1 Čas strávený každým použitím

Při pročítání diskuzí na internetu jsem získal spíše skeptický názor na povědomí veřejnosti o NM, a tedy očekávám slabou vůli uživatelů je používat. K dosažení výše zmíněného hlavního cíle je ale třeba co největšího počtu uživatelů. Faktorem, který na něj má klíčový vliv, je cena, jakou uživatelé platí za používání aplikace ve chvíli, kdy mají aplikaci připravenou k použití, tedy staženou a nainstalovanou. Jedná se především o uživatelův čas strávený za účelem vložení NM do dokumentu. Proto by mělo být potřebné úsilí uživatelů vyvíjené k používání aplikace v poměru ke svému efektu (použití aplikace autorem má větší vliv než použití čtenářem) co nejnižší.

7.1.2 Složitost stažení a instalace

Noví uživatelé nemusí být ochotni aplikaci na svém zařízení zprovoznit, pokud vyžaduje stažení nebo instalaci. Možnost snadného a rychlého použití aplikace bez těchto překážek

by měla zvýšit počet uživatelů, kteří si aplikaci vyzkouší, a důsledkem toho přispět k šíření aplikace.

Odpor vůči stahování a instalacím nových nástrojů je očekáván hlavně u skupiny čtenářů stránek. Ochotu autorů textů a HTML dokumentů zřejmě podporuje jejich víra, že jejich produkt bude číst více lidí a že pro ně všechny bude jejich text čitelnější.

7.1.3 Chronologický vztah vložení NM a publikace

Některé typy uživatelských prostředí umožňují do dokumentu HTML vkládat NM i poté, co je HTML dokument napsán a stránka vystavena na veřejnost. Možností NM vkládat i ve chvíli, kdy je stránka vykreslována čtenáři, disponují prohlížeče. Výhoda spočívá v tom, že o NM mohou být obohaceny také stránky, které již byly vytvořeny bez NM.

7.1.4 Typ uživatelů

Záleží mi na tom, kolik textu obohaceného o NM přinese užívání aplikace průměrným uživatelem.

Užívá-li ji autor, NM jsou vkládány do jeho HTML dokumentů či textů, a žádný čtenář tak již nemusí nechávat NM vkládat. Pokud aplikaci užívá čtenář, o NM budou obohaceny všechny stránky, jež navštíví.

Porovnáván je počet řádků, který přečte průměrný uživatel čtenář, s počtem řádků od průměrného autora-uživatele, které v součtu přečtou všichni čtenáři. Jelikož přesné vyhodnocování vyžaduje statistiku, která není k dispozici, byl použit odhad.

Předpokládá se, že si po sobě každý autor svoji práci čte. Tudíž kterýkoli text je vícekrát přečten nežli napsán. Jedno použití aplikace autorem tak jistě přináší větší množství přečteného obohaceného textu než jedno použití aplikace čtenářem. Tento poměr odhaduji na 100 : 1.

Naproti tomu stojí další předpoklad: průměrný uživatel autor nebude aplikaci používat tak často jako průměrný uživatel čtenář, a tedy frekvence použití aplikace bude u průměrného uživatele čtenáře větší než u průměrného uživatele autora. Tento poměr odhaduji na 60 : 1.

Podle těchto odhadů, pokud je aplikace určena pro užívání autory, představuje to z hlediska efektivity určitou výhodu. Navíc je v zájmu autorů čtenářům poskytovat text již obohacený o NM, kdežto většina čtenářů se s absencí NM smíří mnohem snáze a s o to menší pravděpodobností bude aplikaci vyhledávat.

7.1.5 Nutnost připojení k internetu

Toto kritérium je uvedeno spíše jen pro úplnost. Dnes je již připojení k internetu pro uživatele samozřejmostí. Navíc, stejně jako čtenáři, i autoři internetových stránek potřebují mít v nějaké fázi své činnosti přístup k internetu.

7.1.6 Odlišnost od konkurence

Aby se výsledek této bakalářské práce vyhnul konkurenci v podobě nástrojů popisovaných v kapitole 3, v Existujících řešeních na straně 17, měla by být vzniklá aplikace jiná než ony, a pokud je to možné, takového formátu, který by umožňoval snazší použití.

7.2 Typy aplikací

Možností, jakou by mohla mít výsledná aplikace povahu, je více. Proto jsou zde jednotlivě rozepsány ty, které spadají v úvahu.

7.2.1 Klientská či serverová webová aplikace

Jde o aplikaci dostupnou z internetových prohlížečů. Klientská a webová aplikace (logika programu zpracovávána u uživatele, nebo na vzdáleném serveru) mají podobné vlastnosti.

7.2.1.1 Výhody

- Není ji nutné stahovat ani instalovat, stačí webový prohlížeč a odkaz, což umožňuje snadné sdílení aplikace.
- Předpokládá se, že uživatelé budou (pouze) autoři dokumentů HTML.

7.2.1.2 Nevýhody

- Uživatele zdrží, než aplikaci načtou a text zkopírují např. do textového pole a zpět do preferovaného prostředí.
- NM je nutné vložit před publikací dokumentu.
- Připojení k internetu je nutné při načítání stránky s aplikací.
- Konkurenci tvoří již existující   replacer.
- Navíc v případě serverové varianty je nutná údržba serveru.

7.2.2 Desktopová aplikace nebo spustitelný program

Tento druh SW vyžaduje instalaci na uživatelem spravovaném úložišti a běží na jeho operačním systému.

7.2.2.1 Výhody

- Aplikaci stačí jednou získat, poté již není potřeba přístup k internetu
- Předpokládá se, že uživatelé budou (pouze) autoři dokumentů HTML, jelikož čtenáři si nebudou chtít nic instalovat.

7.2.2.2 Nevýhody

- Použití je rychlejší než u webové aplikace v závislosti na výkonu počítače. Uživatele však přeci jen pozdrží, než aplikaci otevřou a zkopírují svůj text na její vstup a výsledný text zpět do užívaného prostředí.
- Je nutné aplikaci nalézt na internetu a stáhnout, nebo přinést do zařízení jiným způsobem.
- NM je nutné vložit před publikací dokumentu.
- Žádná konkurence v podobě desktopové aplikace nebyla nalezena. Očekává se však, že konkurencí jsou obecně všechna dostupná řešení, jelikož stahovat i používat desktopovou aplikaci jen kvůli vkládání NM bude nejspíše chtít jen málokdo, pokud aplikace nebude ve své funkcionalitě vynikat.

7.2.3 Rozšíření do prohlížeče

Různé prohlížeče mají své obchody s rozšířeními, kde lze nalézt přídavné funkce a nástroje pro práci s webovými stránkami. Prohlížeče jako Google Chrome, MS Edge, Opera a Brave jsou založeny na stejném kódovém základu, Chromiu. Rozšíření kompatibilní s některým z těchto prohlížečů tudíž bývají využitelná všemi prohlížeči této skupiny.

7.2.3.1 Výhody

- Nabízí snadné a rychlé použití přímo v místě, kde je to potřeba.
- Relativně snadné stažení a instalace prostřednictvím obchodu s rozšířeními pro daný prohlížeč. Aplikace však funguje pouze pro daný prohlížeč či skupinu prohlížečů podporujících stejné rozhraní pro rozšíření.
- Funkcionalitu lze využít jak autory textů bez HTML při psaní v prohlížeči i čtenáři při vkládání NM na navštívené stránky, tudíž funguje i pro stránky již publikované.
- Připojení k internetu není nutné při vkládání NM, ale vzhledem k povaze aplikace je při jejím používání uživatel k internetu většinou nucen být připojen.
- Nebylo nalezeno rozšíření s odpovídající funkcionalitou, které by aplikaci tohoto typu konkurovalo.

7.2.3.2 Nevýhody

- Je sice využitelné čtenáři, pisateli novinových článků ve webových redakčních editorech a příspěvků na různých fórech nebo těmi, kteří k psaní HTML dokumentů využívají nějakou webovou aplikaci, avšak autoři webových stránek by o aplikaci zůstali ochuzeni.

7.2.4 Jazyková knihovna

Jazykové knihovny jsou implementace předdefinovaných rozhraní, jež programátoři mohou využít při psaní svých kódů. Tyto knihovny bývají specifické pro určitý programovací jazyk.

7.2.4.1 Výhody

- Nabízí flexibilní použití tam, kde je potřeba – kvůli použití se nemusí zapínat nová aplikace.
- Přístup k internetu je třeba jen pro stažení knihovny.
- Nebyla nalezena konkurenční knihova poskytující odpovídající funkcionalitu

7.2.4.2 Nevýhody

- Vyžaduje import do každého projektu, kde má být použita a čas uživatele, aby se ji naučil používat.
- Je nutné ji pro vložení NM využít před publikací stránky.
- Využít ji mohou pouze autoři webových stránek píšící v jazyce s knihovnou kompatibilním.

7.2.5 Rozšíření editoru HTML nebo IDE

To jsou doplňkové kódy, které vylepšují a přizpůsobují tyto nástroje pro vývoj webových stránek. Nabízí totiž přídavné funkce a nástroje podobně jako rozšíření do webových prohlížečů. Narozdíl od nich tyto slouží výhradně uživatelům daných editorů/IDEs, tedy autorům webových stránek.

7.2.5.1 Výhody

- Snadné použití dostupné přímo z editoru, kde je potřeba vložit NM.
- Často relativně snadné stažení a instalace prostřednictvím obchodu s rozšířeními pro daný editor nebo IDE.
- Připojení k internetu je potřeba pouze pro stažení.
- Nebyla nalezena žádná konkurence v podobě rozšíření do IDE Visual Studio Code ani na JetBrains Marketplace pro produkty WebStorm[49, 50]

Hodnoticí kritérium	Webová aplikace	Stahovaná aplikace	Rozšíření do prohlížeče	Jazyková knihovna	Rozšíření do editoru/IDE
Čas potřebný k použití	Zlý	Značný	Krátký	Krátký	Krátký
Složitost přípravy	Žádná	Značná	Nízká	Značná	Nízká
Vložení NM i po publikaci	Nelze	Nelze	Ano	Nelze	Nelze
Typ uživatelů HTML	Autoři	Autoři	Čtenáři	Někteří autoři	Někteří autoři
Vyžaduje internet	Ano	Zpočátku	Zpočátku	Zpočátku	Zpočátku
Konkurence	Přímá	Nepřímá	Nenalezena	Nenalezena	Nenalezena

Table 7.1: Přehled výhod a nevýhod možných typů aplikace

7.2.5.2 Nevýhody

- Je nutné ho pro vložení NM využít před publikací stránky.
- Je využitelný pouze autory webových stránek používajícími daný nástroj.

7.3 Rozhodnutí o typu aplikace

Rozhodnutí o typu aplikace má významný dopad na další vývoj. Pro lepší přehled jsou proto diskutované výhody a nevýhody jednotlivých typů aplikace ilustrovány v tabulce 7.1.

Desktopovou aplikaci a spustitelný program z výběru vyřazuje nutnost stažení a náležitého spouštění při každém použití. Webová aplikace, již si uživatelé nemusí stahovat a jejíž volba nepřináší podstatné nevýhody, je považována za lepší řešení. Není prospěšné tvořit něco, co už vytvořil někdo jiný, jen, aby to fungovalo jinak. Mnohem větší lze nalézt ve spolupráci na vývoji takového produktu, jenž by mi byl jinak konkurencí. Jednou ze schůdných cest je tedy podílet se na vývoji aplikace   replacer. Nicméně vzhledem k zadání této bakalářské práce bude vyvíjen nový SW.

V případě vývoje knihovny je nutné zvolit jazyk nebo jazyky, pro něž bude psána. Výsledkem rozhodnutí by byl některý z těch nejběžněji používaných pro psaní webových stránek. Použití by však spočívalo ve čtení dokumentace a volání příslušných funkcí v kódu, což zásadně prodlužuje čas potřebný k používání SW. Tyto nepříjemnosti jsou považovány za dostatečně odrazující pro potenciální uživatele, a tedy za dostatečný důvod, proč raději upřednostnit tvorbu rozšíření do editoru, IDE nebo do prohlížeče.

V případě volby rozšíření do prohlížeče by byl zvolen vývoj funkcionality pro čtenáře, jelikož, narozdíl od té pro autory, odpovídá zadání. Autoři textů, např. článků, na internetu

v prostředí webových prohlížečů totiž většinou nepracují s kódem. Nicméně o funkcionalitu pro tyto autory by bylo možné rozšíření později obohatit.

Rozhodnutí, zda je lepší vyvíjet rozšíření do editoru/IDE, nebo do prohlížeče, bylo nakonec učiněno na základě míry naplňování ideálního stavu. Výhodou aplikace, která alespoň částečně naplňuje ideální stav, je menší důvod nahrazovat ji v budoucnu jiným řešením. V tomto kontextu je stav považován za ideální, když mají všechny webové stránky správně vložené NM, a prohlížeče čtenářů je tak nemusí pokaždé vkládat, když stránku načítají. Proto bylo rozhodnuto vyvíjet rozšíření pro IDE či textový editor.

7.3.1 Volba IDE / textového editoru

V úvahu připadají nejpoužívanější nástroje pro psaní dokumentů HTML, tedy VS Code a ty od společnosti JetBrains.

Všechna integrovaná vývojová prostředí od společnosti JetBrains mají stejnou množinu pluginů. Na svých stránkách uvádí, že mají téměř 16 milionů uživatelů.[51] Naproti tomu VS Code jich podle dostupných informací měl začátkem roku 2021 kolem 14 milionů.[52, 53]. Vzhledem k tomu, že se tato čísla přibližně rovnají a k dispozici nejsou bližší informace o využívaní těchto nástrojů pro vývoj HTML dokumentů, bylo zvolen nástroj VS Code, který se jako textový editor spouští rychleji a má obecně menší nároky na výkon počítače.

Kapitola 8

Uživatelská konfigurace

Noví uživatelé budou moci využívat implicitní nastavení. Jak je psáno v kapitole 2 na str. 10, některá, zde tzv. hrubá, pravidla však nelze s přiměřenou náročností aplikovat tak, aby bezchybně fungovala pro každý text. Následující odstavce se tedy zabývají možnostmi, jak si uživatelé budou moci aplikaci hrubých pravidel podle svého uvážení přizpůsobit, tak aby co nejvíce vyhovovala jejich textům.

Aplikace není navrhována pro uživatele čtenáře, u kterých se vůle nastavovat optimální chování pro každou načtenou stránku neočekává, nýbrž pro uživatele autory. Budou tak vyžadovány možnosti jejího podrobného konfiguračního nastavení, kterých by jim mělo být nabídnuto co možná nejvíce.

8.1 Pravidla pro vkládání NM

Jelikož budou pravidla pro vkládání NM implementována pomocí regulárních výrazů, je na nich založena také jejich konfigurace. Pravidla, která lze částečně implementovat, jsou v této bakalářské práci nazvány hrubými pravidly. Ta jsou zde opět rozdělena – tentokrát na dvě skupiny.

8.1.1 Hrubá pravidla automatická

Tato pravidla lze jednoduchou volbou přepnout. Program bude mít připravenou skupinu regulárních výrazů, které zůstávají vypnuty, popř. zapnuty, dokud uživatel nezvolí jinak.

Jedná se o tato pravidla:

- Krátké předložky a slabičné spojky
- Kalendářní data
- Tituly před jmény (základní množina)
- Matematické operátory

- Pomlčky a znaky minus
- Lomítka
- Matematické závorky

8.1.2 Hrubá pravidla manuální

Konfigurace pro druhou podskupinu hrubých pravidel je pro uživatele složitější. Ten musí do seznamu doplnit regulární výrazy a případně označit mezery, namísto nichž se vkládají NM, jelikož je aplikace předem znát nebude. Za vzniklou chybovost pak odpovídá on. Jedná se o pravidla, která vyžadují např. příliš mnoho regulárních výrazů na to, aby byla aplikována dokonale. Tyto výrazy jsou často specifické pro daný text.

Jedná se o tato pravidla:

- Čísla a značky
- Vyjmenované následované zkratky
- Složené zkratky
- Tituly před jmény (uživatelem přidané)

8.2 Rozhodnutí o další konfiguraci

Další nastavení, která mohou být užitečná, se nemusí přímo týkat NM, avšak je nutné rozhodnout, zda nemá smysl je přesto implementovat. Proto jsou diskutována v této sekci.

8.2.1 Automatické členění dlouhých čísel

Členění čísel (nezlomitelnými) mezerami není zvažováno ani jako volitelná funkce, jelikož s sebou nese přílišné komplikace. Bylo by například potřeba odlišit přídavná jména zapisovaná číslem a značkou od ostatních čísel. Takový výraz se totiž píše bez mezery a číslice se nečlení.[9] Navíc *false positives*, které lze od této funcionality očekávat, může mást uživatele.

8.2.2 Nahrazování spojovníku pomlčkou

Při vkládání mezer se bude program na některých místech, kde je psán spojovník, rozhodovat, zda by nemělo jít o pomlčku (spojovník nebývá obklopen mezerami). Z toho důvodu se tedy nabízí spojovníky obklopené mezerami nahrazovat znakem pomlčky tam, kde je k tomu důvod. Bylo však rozhodnuto, že spojovník pomlčkou program nahrazovat nebude, a to mj. proto, že by to chování aplikace komplikovalo jak z pohledu vývoje, tak pro uživatele.

8.2.3 Zápis vkládaných NM

Pokud uživatel potřebuje jiný zápis NM než implicitní (viz kapitola 6, strana 28), bude mít možnost zvolit, jakými znaky budou mezery nahrazovány, podobně, jako je to umožňováno v programu Vlna.[16] Nicméně pro jednoduchost použití bude narozdíl od Vlny tato aplikace nabízet uživateli množinu možností, a ten si tak nebude moct zvolit zcela libovolný zápis. V nabídce budou všechny entity pro psaní NM psané se středníkem (více viz kapitola 6, str. 27).

8.2.4 Přepisování původních NM

Pokud si uživatelé nejsou jisti mezerami, jež jsou v dokumentu obsaženy, využijí přepínač, který programu dovoluje nahrazovat již přítomné NM. Tam, kde by NM nevkládal, bude ve výsledku vždy klasická mezera bez ohledu na to, zda tam původně byla NM, či klasická mezera, třebaže byla napsána entitou.

8.3 Závěr

Konfigurovatelnost aplikace pomocí RV umožňuje dostatečnou flexibilitu a je řešením pro odstranění většiny nedokonalostí v oblasti aplikace pravidel pro zalamování řádků. Základními konfiguračními volbami je přepisování původních NM a volba zápisu vkládaných NM. Ostatní možnosti jsou považovány spíše za pokročilá nastavení, u nichž se nepředpokládá, že budou často potřebné.

Part II

Realizace

Kapitola 9

Analýza

Kapitola Analýza stanovuje cíle projektu a rozkládá je na jednotlivé požadavky. Ty stanovují rámec a rozsah celé aplikace a poskytuje opěrné body při tvorbě jejího návrhu a během vývoje.

9.1 Cíle

Výsledkem tohoto projektu má být aplikace usnadňující úpravu dokumentů HTML. Při vytyčování níže uvedených cílů vycházím mj. z informací a úvah popsaných v různých částech této bakalářské práce.

BG 1 – Efektivita úpravy dokumentů HTML

Jako autor bakalářské práce potřebuji umožnit automatické vkládání NM do dokumentů HTML jejich autorům, protože manuální vkládání je náročné na jejich čas a pozornost, ale zároveň přispívá k čitelnosti textu.

BG 2 – Počet stažení

Jako autor bakalářské práce chci za 60 dnů od prvního vydání aplikace dosáhnout alespoň 15 instalací, protože chci vědět, že má práce má praktické využití.

BG 3 – Povědomí o pravidlech pro zalamování řádků

Jako autor bakalářské práce chci rozšířit povědomí o existenci jazykových a typografických pravidlech pro zalamování řádků alespoň mezi 30 lidmi ze svého okolí anebo uživatelů aplikace, protože obecně nejsou příliš známá a aplikovaná ačkoli jejich aplikace zlepšuje čitelnost téměř jakéhokoli textu a tyto znalosti u autorů dokumentů HTML navíc zvyšují šanci, že aplikaci vyhledají či použijí.

9.2 Procesní diagram tvorby dokumentu

Na obrázku 9.1 je vidět diagram procesu tvorby dokumentů HTML při používání aplikace pro automatické vkládání NM.

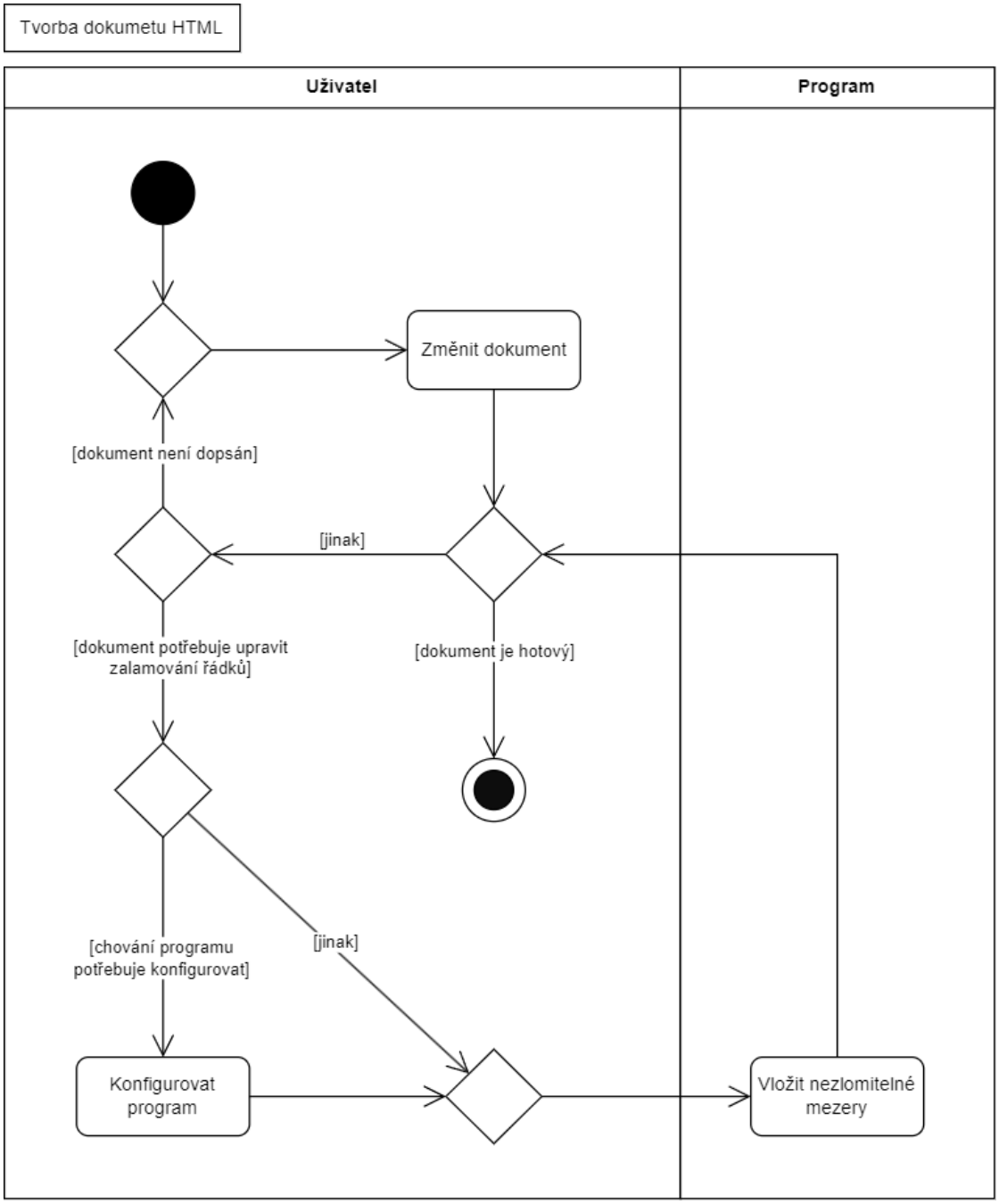

Figure 9.1: Procesní diagram tvorby dokumentu s pomocí vyvíjeného rozšíření

Diagram je užitečný pro lepší představu, jak bude práce s HTML dokumentem a vyvíjeným rozšířením probíhat. K tomuto procesu se vztahují některé požadavky (viz další strana).

9.3 Požadavky

Níže jsou formulovány požadavky na základě konzultací s vedoucím a podle výstupů z předchozích kapitol.

Jejich číslování vypovídá o tom, z jakého cíle byly vytvořeny. Je totiž ve formátu *BRQ <číslo cíle>.<číslo business požadavku v rámci cíle>*. Jelikož se, jak je rozhodnuto v kapitole 7 o typu aplikace na str. 35, bude jednat o rozšíření nástroje VS Code, uživateli jsou zde míněni uživatelé tohoto textového editoru.

9.3.1 Business požadavky

BRQ 1.1 – Úprava dokumentů vkládáním NM

Uživatelé potřebují možnost nechat vložit do svého dokumentu NM podle českých jazykových a typografických pravidel, aby jejich dokumentům nechyběla řádná úprava.

BRQ 1.2 – Oprava přítomných NM

Uživatelé potřebují možnost před vkládáním NM nechat program, aby měnil také již přítomné NM na standardní mezery, pokud o nich mají pochyby, aby mohly být opraveny chyby způsobené např. chybným manuálním vkládáním nebo nesprávnou konfigurací při posledním běhu programu.

BRQ 1.3 – Sjednocení zápisu NM

Uživatelé potřebují možnost při vkládání nechat program nahrazovat všechny zápisy přítomných NM v textu zápisem zvoleným pro vkládané NM, aby byl jejich zápis snadno automaticky sjednocen.

BRQ 1.4 – Spolehlivost implicitního nastavení

Uživatelé potřebují, aby aplikace dobře fungovala už v implicitním nastavení, aby mohli funkcionalitu snadno a rychle vyzkoušet a přitom se často nemuseli zabývat konfigurací.

BRQ 1.5 – Konfigurace chování

Uživatelé potřebují možnost přizpůsobit si chování programu při vkládání NM do jejich dokumentu, aby byla úprava jejich textů co možná nejlepší.

BRQ 1.6 – Průvodce pravidly

Uživatelé potřebují mít možnost přečíst si, jaká pravidla program dovede aplikovat, aby mohli lépe posoudit složitost jejich případné konfigurace a její přínosy pro úpravu jejich textu.

BRQ 1.7 – Volba zápisu NM

Uživatelé potřebují konfigurovat zápis programem vkládaných NM, aby odpovídaly jejich návykům a byly kompatibilní s jinými prostředími po případném přesunutí textu.

BRQ 1.8 – Zřetel na čas uživatelů

Uživatelé potřebují, aby je vkládání NM nezdržovalo a aby při něm aplikace neodváděla pozornost od práce na HTML dokumentu.

BRQ 1.9 – Dodatečná vylepšení

Uživatelé potřebují místo, kam mohou psát zpětnou vazbu, a aby aplikace byla dále vyvíjena i po svém nasazení, protože časem uvidí nějaký prostor pro její vylepšení a budou se chtít o svůj nápad podělit.

BRQ 1.10 – Zpětná vazba pro vývojáře

Jako vývojář aplikace k jejímu fungování od uživatelů potřebuji dostávat zpětnou vazbu, abych na jejím základě mohl aplikaci vylepšovat.

BRQ 2.1 – Přítomnost rozšíření ve VS Marketplace

Uživatelé potřebují vyhledávat a stahovat aplikaci jako rozšíření přímo ve VS Marketplace, protože právě tam k ní mají nejsnazší přístup. [54]

BRQ 2.2 – Popis funkcionality

Uživatelé potřebují mít přístup k popisu funkcionality aplikace v českém a anglickém jazyce ještě před jejím stažením, jelikož na jeho základě budou její stažení a používání teprve zvažovat.

9.3.2 Systémové požadavky

V této části jsou funkční požadavky vyjádřeny diagramem případů užití na obrázku 9.2. Ostatní (tzv. nefunkční) systémové požadavky jsou vypsány níže.

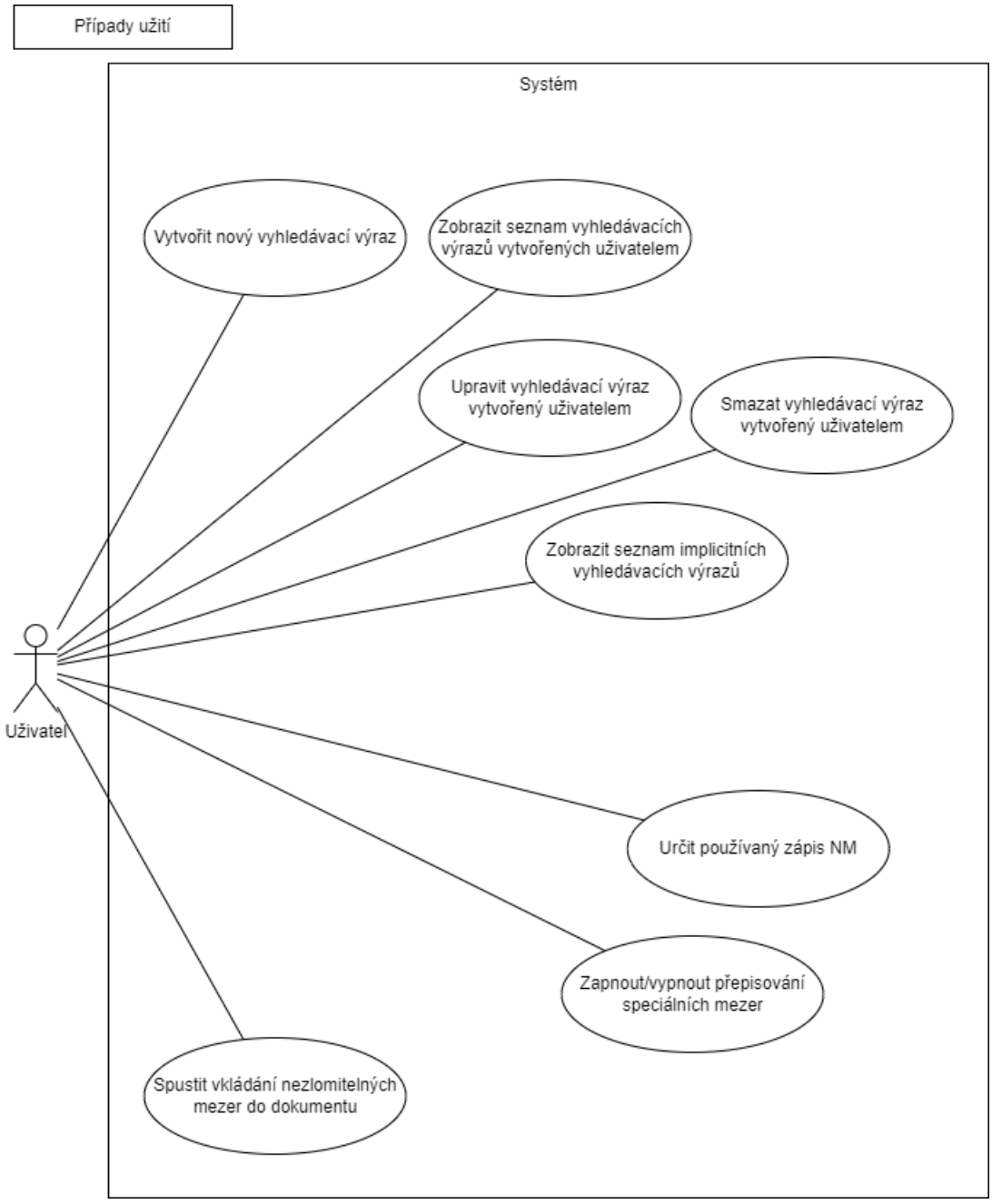

Figure 9.2: Procesní diagram tvorby dokumentu s pomocí rozšíření

Seznam nefunkčních požadavků je číslován takto: *SRQ <číslo cíle>.<číslo business*

požadavku, ze kterého nefunkční požadavek vychází>.<číslo nefunkčního požadavku>.

SRQ 1.8.1 – Forma aplikace

Vkládání NM nebude průměrně trvat déle než pár vteřin.

SRQ 2.1.1 – Forma aplikace

Aplikace bude ve formě rozšíření pro nástroj VS Code.

SRQ 2.2.1 – Distribuce

Aplikace bude vydána na VS Marketplacu.

SRQ 2.2.2 – Veřejný popis

K aplikaci bude popis její funkcionality dostupný pro uživatele navštěvující stránku s aplikací. (Jde o požadavek tranzitivně vznikající z toho, že aplikace má být vydána na VS Marketplacu, kde je soubor readme(.md) více než užitečný, jelikož je jím rozšíření prezentováno.[54, 50])

9.3.3 Závěr analýzy

Projekt lze rozdělit na dvě části, samotné rozšíření nástroje VS Code a šíření informací o NM. Pro rozšíření, první zmíněnou část, jsou stěžejními požadavky funkcionalita vkládání NM a dostačující spolehlivost implicitní konfigurace. Druhou část lze realizovat prostřednictvím prezentační stránky rozšíření, jež je definována souborem readme.md. Tu uživatelé uvidí, když v obchodě pomocí nástroje VS Code či webového prohlížeče aplikaci najdou a rozkliknou.[54, 50]

Jelikož společnost Microsoft nabízí veřejně dostupné návody pro vývojáře rozšíření nástroje VS Code, tyto návody jsou psány hlavně pro TypeScript a taková rozšíření jsou v něm obvykle psána, bude tento programovací jazyk zřejmě nejschůdnější cestou.[55]

Kapitola 10

Návrh

Tato kapitola se zaměřuje na návrh architektury SW. Definuje se zde, jak budou jednotlivé komponenty systému izolovány a jak na sobě budou navzájem závislé, jak spolu budou komunikovat. V tomto návrhu architektury bude brán zřetel hlavně na udržitelnost a možnost recyklace kódu. Cílem je vytvořit systém tak, aby se dokázal přizpůsobit budoucím potřebám, kterými může být např. migrace na jinou platformu než VS Code nebo do jiného kontextu než HTML.

10.1 Diagramy komponent

Na diagramech komponent je graficky znázorněna architektura aplikace. Dále jsou v této kapitole obsaženy popisy jednotlivých součástí systému a úloh, které v systému plní.

Aplikace je členěna tak, aby bylo snadné aplikaci implementovat i do jiného prostředí než nástroje VS Code. I to je zachyceno v diagramech komponent, a to dvěma barvami.

Modrou barvu mají v diagramech ty části, které jsou přímo závislé na prostředí. V tomto případě tedy na VS Code Extension API. V terminologii návrhových vzorů se tedy jedná o adaptéry.

Zelenou barvou jsou pak značeny komponenty, které by i v případě přetvoření aplikace do jiné podoby – např. do formy rozšíření do prohlížeče či jiného textového editoru než VS Code.

10.1.1 Přehled architektury

Na následujícím obrázku 10.1 je diagram zachycující celkový pohled na architekturu systému. Jedná se tedy přehledné vyobrazení všech částí aplikace, jejich závislostí a poskytovaných rozhraní. Podrobnější popis jednotlivých komponent je obsažen v následujících odstavcích.

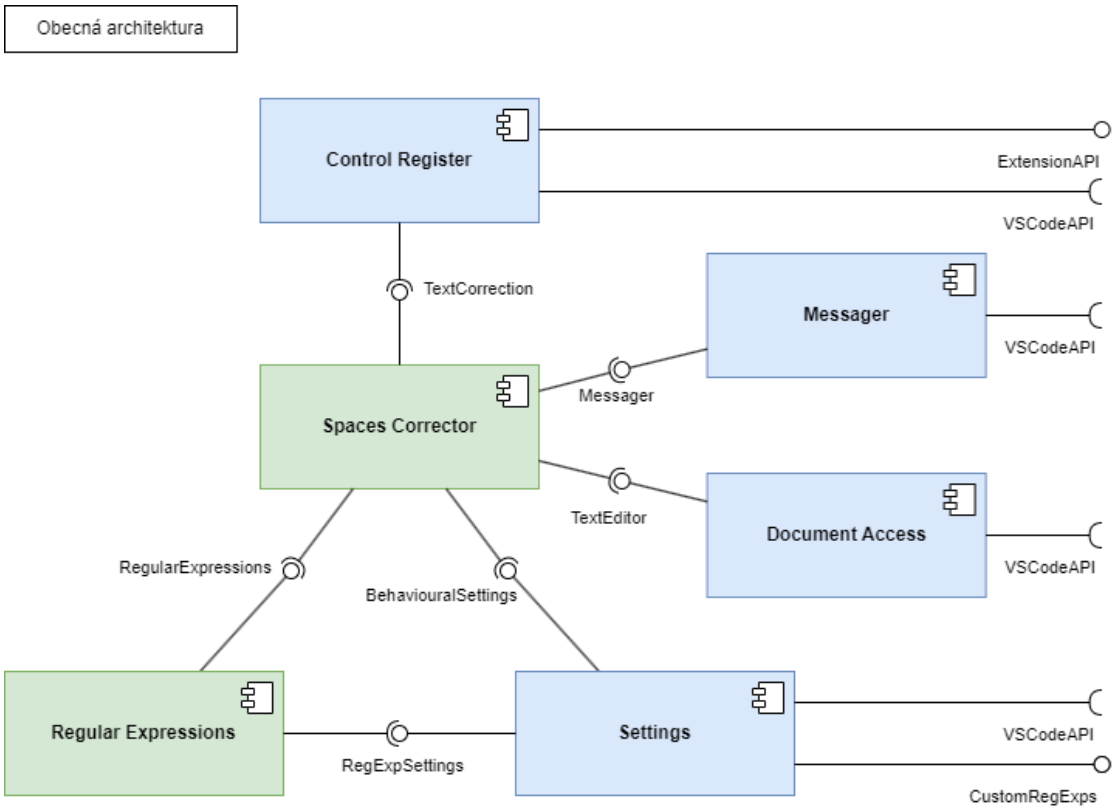

Figure 10.1: Přehledový diagram komponent aplikace

10.1.2 Architektura části Control Register

Control Register registruje ovládací prostředky (viz následující odstavec) a navazuje je na proces vkládání nezlomitelných mezer.

V kontextu VS Code se funkcionalita rozšíření spouští prostřednictvím tzv. příkazů. Příkaz může být v programu VS Code spuštěn následujícími způsoby (ovládacími prostředky):

- prostřednictvím volby příkazů v tzv. "paletě příkazů",
- klávesovou zkratkou, pokud je nastavena.

Aby bylo možné příkaz najít v paletě příkazů, úkolem Control Registeru je namapovat příkaz na funkci celého procesu s dokumentem.

To vykonává komponenta Extension Activator. Její název je odvozen od funkce activate, kterou implementuje Extension API, jehož voláním se rozšíření aktivuje, tedy si připravuje data pro své potřeby. Na obrázku 10.2 je vidět, jak propojuje funkcionalitu s příkazem a s implicitní klávesovou zkratkou pomocí komponent Command Register a Keybinding Register.

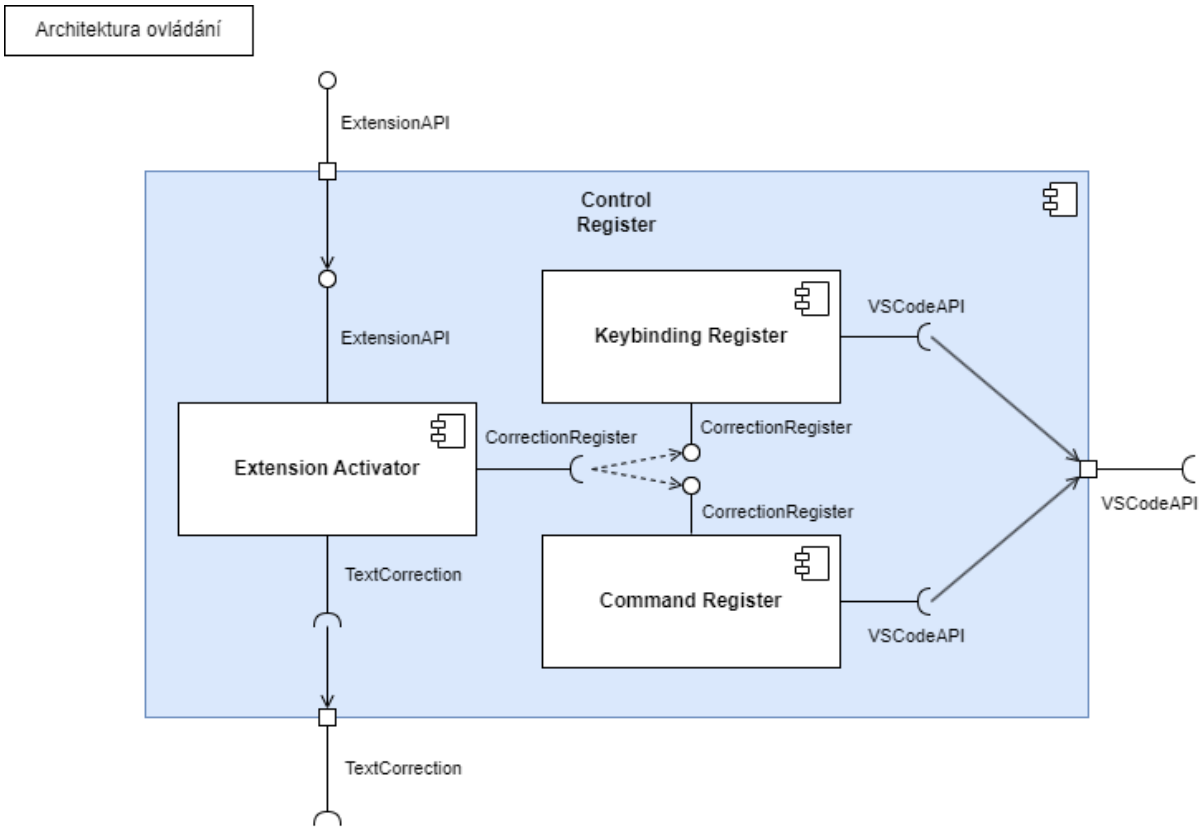

Figure 10.2: Diagram části Control Register

10.1.3 Architektura části Spaces Corrector

Veškerou práci s nezlomitelnými mezerami v dokumentu vykonává komponenta Spaces Corrector (viz obrázek 10.3).

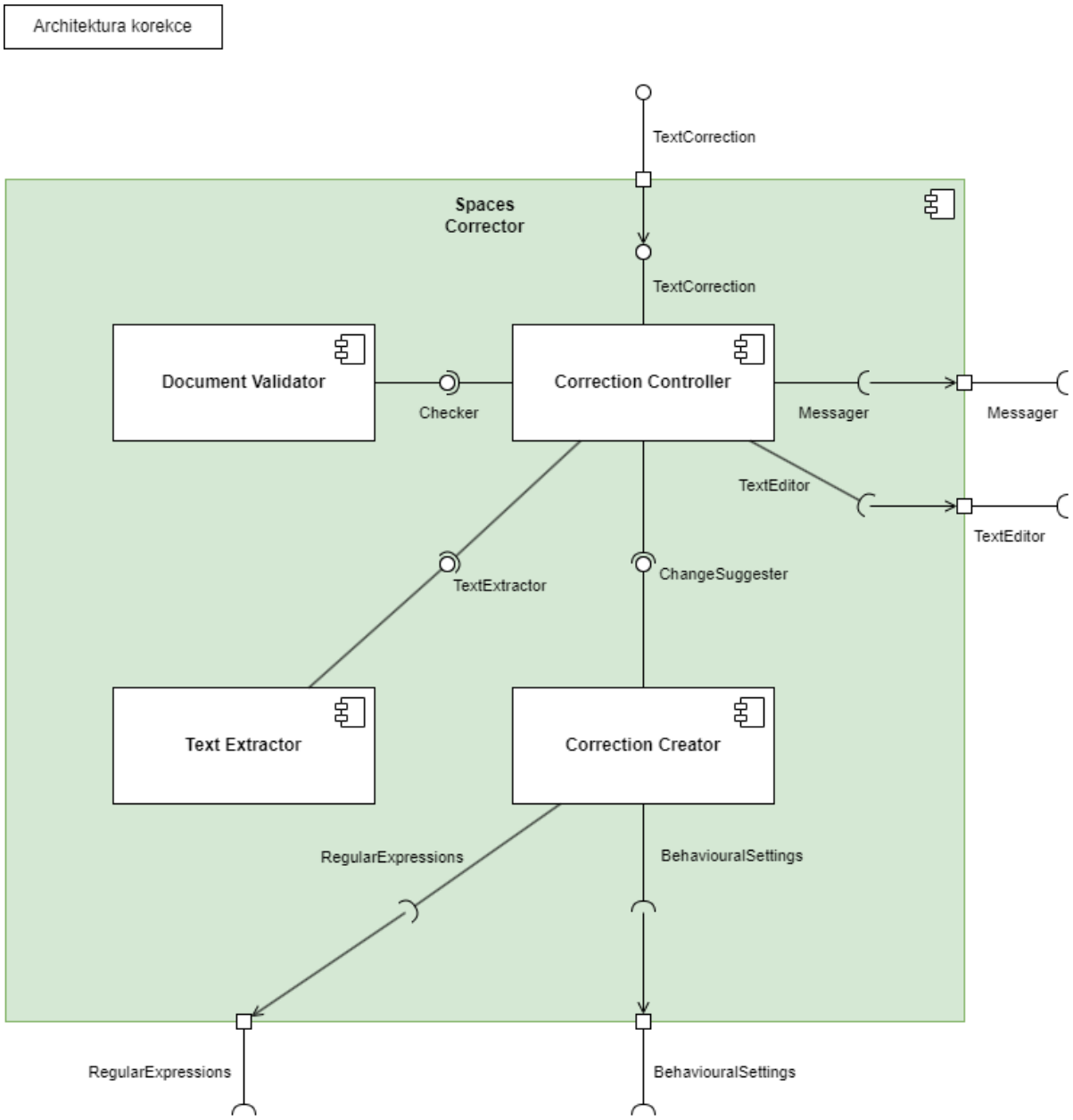

Figure 10.3: Diagram části Spaces Corrector

Práci deleguje Correction Controller ostatním komponentám v tomto pořadí:

1. adaptér Text Editor je využit k přečtení dokumentu,

2. Document Validator zkontroluje, zda je v dokumentu validní notace HTML a zda je text v něm psán česky,
    - Pozn.: Pokud je zde zjištěno, že dokument nesplňuje nutné předpoklady, je operace pozastavena a Messager je pověřen upozornit na to uživatele.
3. Text Extractor z dokumentu odfiltruje vše nepotřebné, a získá tak jednotlivé textové pasáže, na nichž provede operaci Correction Creatoru,
    - Pozn.: Text Extractor se přitom stará pouze o získání textových pasáží a provádění libovolné operace na nich. Tuto operaci Correction Creatoru mu předává opět delegující Correction Controller.
4. Text Editor je opět využit, tentokrát však k zápisu změn do dokumentu,
5. Messager oznámí uživateli úspěšné dokončení operace

Chování Correction Creatoru závisí na konfiguraci. Ta je čtena prostřednictvím komponenty Settings. Vyhledávané regulární výrazy pro přepis nezlomitelných mezer jsou čteny z komponenty Regular Expressions.

10.1.4 Architektura části Regular Expressions

Jak naznačuje obrázek 10.4, tato komponenta se skládá ze dvou menších a podobně jako Correction Controller v komponentě Spaces Corrector je zde Regular Expressions Picker tou řídící komponentou.

V databázi jsou regulární výrazy potřebné pro vyhledávání frází v textu, v nichž mají být psány NM, rozděleny do různých skupin (např. polí/kolekcí). Ty slouží pro aplikaci jednotlivých pravidel při různém nastavení pro tato pravidla. Komponentou Regular Expression Picker je vždy vytvořeno sjednocení různých těchto skupin:

1. volitelné skupiny, které odpovídají aktuální konfiguraci,
2. všechny nevolitelné skupiny,
3. skupina regulárních výrazů definovaných uživatelem.

Výsledek v podobě početné skupiny lze číst prostřednictvím rozhraní RegularExpressions.

10.1.5 Architektura části Settings

Větší část uživatelské konfigurace je komponentou Settings Access čtena z rozhraní VSCodeAPI (viz obrázek 10.5). Dále je zde deklarována komponenta předávající množinu regulárních výrazů, které uživatel sám definuje pro vkládání NM.

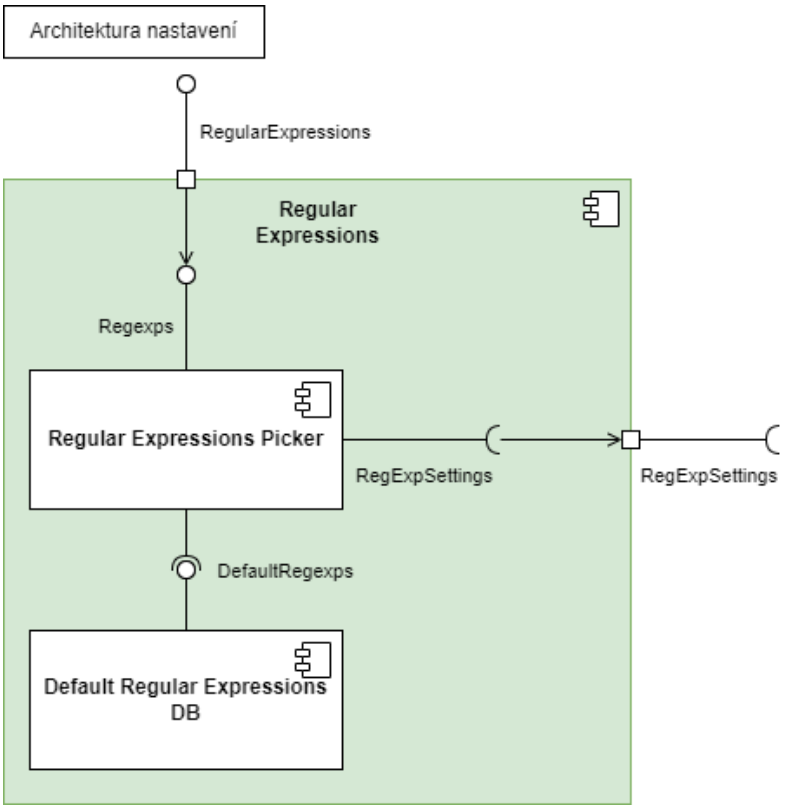

Figure 10.4: Diagram části Regular Expressions

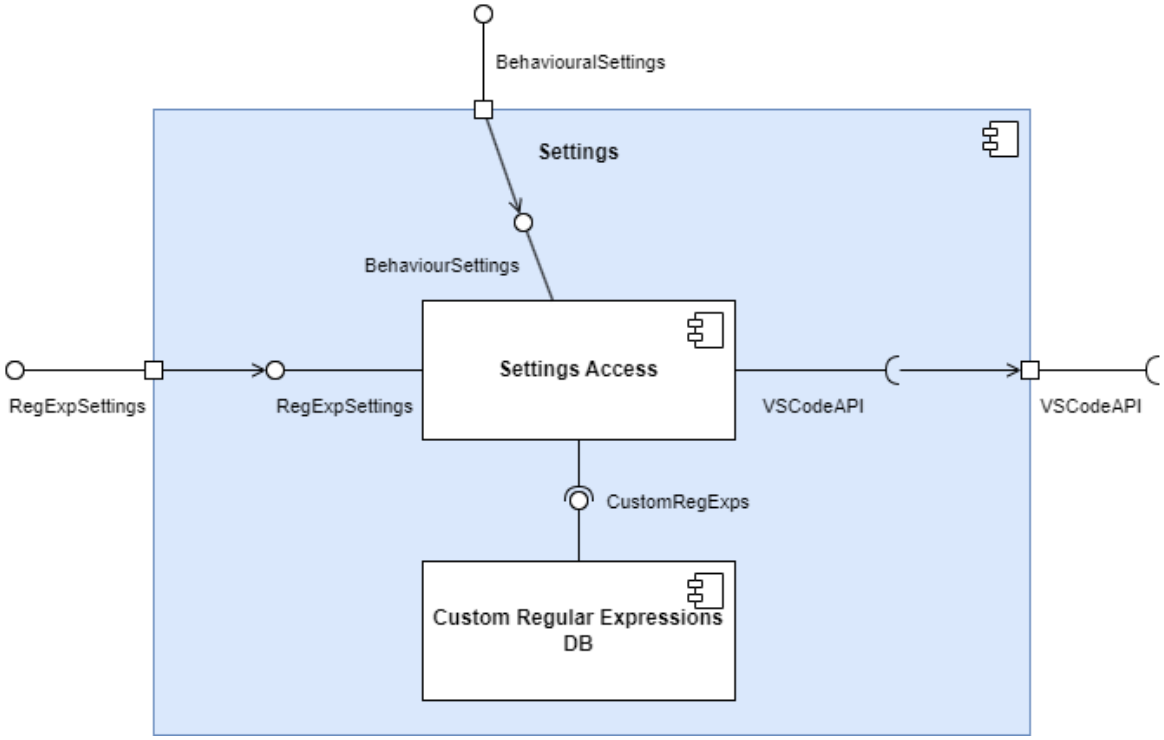

Figure 10.5: Diagram části Settings

10.1.6 Část Document Access

Tato komponenta není dále rozdělena na menší podčásti, a proto zde není její grafický detail. Nabízí API, prostřednictvím něhož aplikace poskytuje uživateli informace o proběhlém vkládání nezlomitelných mezer.

10.1.7 Část Messager

Podobně jako Document Access nebylo nutné dále rozdělit tuto komponentu. Jak název naznačuje, jde o adaptér, který v prostředí VS Code informuje uživatele o dokončení operace.

10.2 Závěr návrhu

Aplikace je navržena tak, aby bylo možné hlavní část její implementace použít i jiném prostředí než VS Code. V podstatě se jedná o tzv. model-view-controller s view rozděleným na izolované části s různou funkcionalitou. [56] Controller funguje na bázi vzoru pipeline. [57]

Ačkoli je aplikace psána jako rozšíření do nástroje VS Code, tento návrh umožňuje snadnou recyklaci kódu v případě její migrace na jinou platformu, např. do prohlížeče nebo do formy desktopové aplikace. View totiž slouží jako množina adaptérů zasazující funkcionalitu do konkrétního prostředí. Zbytek aplikace tudíž může být znovu použit, např. až bude vhodné aplikaci nabídnout také uživatelům jiných nástrojů pro úpravu HTML dokumentů společnosti JetBrains, tedy konkurenci VS Codu.

Přenos aplikace mimo prostředí HTML již vyžaduje změnu logiky controlleru. Se zvolenou architekturou je to možné prostřednictvím změny příslušných filtrů v jeho pipelině, které jsou od sebe izolovány, aby byla taková změna co nejsnazší.

Objeví-li se požadavek podporovat jiné jazyky než český, budou tím zasaženy následující komponenty:

- Regular Expressions, protože regulární výrazy přímo závisí na typografických pravidlech jazyka.
- Settings, z obdobného důvodu.
- Messager, protože hlášky informující o průběhu vkládání je vhodné psát v daném jazyce (resp. v jazyce předvoleném uživatelem).

Kapitola 11

Implementace

Implementace je klíčovou součástí vývoje SW a zahrnuje volbu konkrétních technologií, převod návrhu SW na zdrojový kód a jeho organizaci. Jak bylo již zmíněno, aplikace je napsána v jazyce TypeScript. To umožňuje snadné využití schopností nástroje VS Code prostřednictvím rozhraní VSCodeAPI a zároveň umožňuje recyklaci kódu v případě migrace např. do internetových prohlížečů. Při integraci funkcionality s textovým editorem VS Code bylo postupováno podle návodů dostupných na jeho stránkách.[1]

[1] https://code.visualstudio.com/api/get-started/your-first-extension

11.1 Manifest

Projekt není tvořen pouze zdrojovým kódem. Klíčovou roli v každém rozšíření VS Code plní také manifest package.json.[58] Je umístěn v kořenovém adresáři projektu a obsahuje např. tyto hodnoty:

- name – identifikátor rozšíření,
- version – verze,
- publisher – identifikátor účtu, pod nímž je rozšíření vydáno,
- engines – kompatibilní verze VS Code,
- description – krátký popis,
- categories – pole štítků popisujících, čím se rozšíření zabývá, může být např. "Formatters", "Debuggers" nebo "Snippets",
- contributes – pole definující příkazy, pod kterými je rozšíření aktivováno, a možnosti konfigurace.

Tyto hodnoty jsou nezbytné pro správné zpracování nástrojem VS Code a pro správnou prezentaci rozšíření v obchodě VS Code Marketplace.[59] [58]

11.2 Ovládání

Funkcionalita rozšíření se ve VS Code aktivuje prostřednictvím spuštění vybraného příkazu v tzv. paletě příkazů, klávesovou zkratkou nebo tlačítkem. Pro toto rozšíření byly zvoleny první dva způsoby. Aby byl příkaz dostupný a volal příslušné funkce, je nutné ho mít uvedený v manifestu a, obvykle uvnitř exportované funkce activate, tak zvaně registrovaný.[54]

11.2.1 Soubor extensionActivator.ts

Každé rozšíření nástroje VS Code musí exportovat funkci activate. Ta je spouštěna, když je použit některý z příkazů deklarovaných v manifestu, a v případě této aplikace se nachází v souboru extensionActivator.ts a volá funkci souboru commandRegister.ts pro registraci příkazu.

Obdobně by soubor extensionActivator.ts mohl exportovat také funkci deactivate, která je volána, když je nástroj VS Code vypnut, nebo je rozšíření odinstalováno, či jen vypnuto, slouží tedy k provedení "úklidových" operací.[54, 58]

11.2.2 Soubor commandRegister.ts

Exportovaná funkce registerCommand slouží k navázání příkazu na funkci opravy NM v aktivním dokumentu. V principu vytváří a ukládá posluchač tohoto příkazu.[60]

11.3 Přístup k souboru HTML

Při aktivaci pracuje program vždy s právě otevřeným souborem. Tento soubor je na začátku celého procesu pro vkládání NM přečten a na jeho konci jsou do tohoto souboru zapsány změny (přepisy mezer) vygenerované na základě přečteného textu.

11.3.1 Soubor Change.ts

V souboru je definován datový typ Change, který slouží pro přenos dat mezi komponentami Document Access a Spaces Corrector (data transfer object). Obsahuje informace o změně, která má být v dokumentu provedena: indexy počátku a konce nahrazované části dokumentu a nový řetězec.

11.3.2 Soubor documentAccess.ts

V tomto souboru se nachází funkce pro čtení obsahu tohoto souboru a zápis změn do něj na příslušné indexy.

11.3.2.1 Čtení dokumentu

Při čtení se zároveň inicializuje hodnota pro chystaný zápis (proměnná activeEditor, viz 11.1). To by mohlo být důležité, pokud by byl při čtení aktivní jiný dokument než při zápisu (uživatel by stihl okno přepnout). Vzhledem k rychlosti programu se však nepředpokládá, že by to byla častá hrozba.

```
27
28  let activeEditor: vscode.TextEditor
29  let activeDocument: vscode.TextDocument
30
31  export function read(): string {
32      activeEditor = getActiveEditor()
33      activeDocument = getDocumentFromEditor(activeEditor)
34      return activeDocument.getText()
35  }
36
```

Figure 11.1: Čtení obsahu dokumentu

11.3.2.2 Zápis změn do dokumentu

Na konci zpracovávání příkazu k vložení je komponentou Spaces Corrector volána funkce applyChanges (viz obrázek 11.2) a je do ní předáno pole objektů typu Change k zápisu do dokumentu.

```
33
34  export async function applyChanges(changes: Change[]) {
35      checkRanges(changes, activeDocument)
36
37      await activeEditor.edit((editBuilder) => {
38          let change: Change | undefined
39          while (change = changes.pop()) {
40              rewriteSection(change!, activeDocument, editBuilder)
41          }
42      })
43  }
44
```

Figure 11.2: Zápis změn do dokumentu

Funkce activeEditor.edit je asynchronní. Otevře přístup k úpravám uživatelem aktuálně zobrazeného souboru a postupně provádí veškeré změny čtené z argumentu. Následně přístup zavře.

Konstrukce async-await zajišťuje, že se tento asynchronní kód chová synchronně.[61] Kdyby se kód prováděl asynchronně, mohlo by dojít ke konfliktu mezi dvěma vlákny, které chtějí do dokumentu přistupovat ve stejnou chvíli. Jelikož je tato funkce po jednom zavolání příkazu ke vložení NM volána vícekrát, má smysl se tomuto konkurenčnímu přístupu snažit vyvarovat.

11.4 Zpracování textu

Textem dokumentu se zabývá komponenta Spaces Corrector. Validuje ho, čte konfiguraci, nechává zapsat opravné změny a vypsat výsledek. Strukturu této komponenty lze vidět na obrázku 10.3 na str. 56 v kapitole 10, Návrh.

11.4.1 Soubor correctionController.ts

Komponenta Correction Controller implementována v souboru correctionController.ts nechává měnit text dokumentu celkem až dvakrát. Nejprve když sjednocuje zápis nezlomitelných mezer, následně když vkládá NM. Pokaždé přitom provádí následující:

1. nejprve kontroluje validitu dokumentu,
2. následně ho rozkládá na úryvky textů v jednotlivých elementech,
3. dále k těmto úryvkům generuje příslušné změny – buď změny sjednocující zápis NM v dokumentu, nebo změny pro vkládání NM do dokumentu a jejich případné odebírání z něj,
4. nakonec vygenerované změny nechá zapsat.

Po skončení zápisů uživateli nechá prostřednictvím komponenty Messager vypsat výsledek operace. Vrchní úroveň popisovaného algoritmu je vidět na obrázku 11.3, ta spodní (část vykonávaná dvakrát) na obrázku 11.4.

```
7
8   export async function correctActiveDocument() {
9       try {
10          await changeTexts(createSpaceUnifications)
11          inform("Nalezené výrazy: " + await changeTexts(createAllCorrections))
12      } catch (error) {
13          if (error instanceof Error) alarm(`Je mi líto, něco se nepovedlo: ${error.message}`)
14      }
15  }
16
```

Figure 11.3: Vrchní úroveň algoritmu vkládání NM

```
16
17  async function changeTexts(changesGeneration: (correctedText: string) => Change[]): Promise<number> {
18      let HTMLText = obtainText()
19      let textParts = DOMRecurser.extractTexts(HTMLText)
20
21      const changes: Change[] = generateChanges(textParts, changesGeneration)
22      let numberOfChanges = changes.length
23      await documentAccess.applyChanges(changes)
24      return numberOfChanges
25  }
26
```

Figure 11.4: Spodní úroveň algoritmu – vlastní změna textů

11.4.2 Soubor documentValidator.ts

O validaci dokumentu se stará komponenta Document Validator v souboru documentValidator.ts. Validace se provádí s využitím rozhraní DOMParser, které slouží k vytvoření objektu dokumentu z textu v HTML. Princip je ten, že DOMParser pozná, když text není validní HTML, a se ho lze zeptat na chybu (viz obrázek 11.5).

Jak je v ukázce také vidět, dále se kontroluje, zda je v tagu dokumentu *html* přítomný atribut lang a zda značí češtinu (podrobnosti viz kapitola 4, Rozpoznávání českého textu, str. 21).

```
 9      if (!isParsedOK(documentElement)) return false
10
11      const lang = documentElement.getAttribute("lang")
12      return lang ? lang.startsWith("cs") : true
```

Figure 11.5: Validace textu dokumentu

11.4.3 Soubor correctionCreator.ts

Data potřebná k opravě mezer generuje na základě vyhledávání regulárními výrazy, převážně staženými z DB, kód v souboru correctionCreator.ts komponenty Correction Creator. Jde o indexy a opravné řetězce. Při vyhledávání se zjišťuje index shody a v nalezeném textu se nahradí jeden zápis mezery nějakým jiným.

Podobný algoritmus funguje i při sjednocování zápisů NM v dokumentu. Pro tento účel jsou regulárními výrazy vyhledávány právě různé způsoby zápisů NM (*   * a * * více o zápisu NM v kapitole 6 na str. 27).

11.4.4 Soubor textExtractor.ts

Aby bylo možné pracovat s prostým textem, je nutné ho totiž nejprve z textu dokumentu, který je v jazyce HTML, extrahovat. To obstarává komponenta Text Extractor v souboru textExtractor.ts. (Rešerše vztahující se k této problematice se nachází viz kapitola 14. Práce s členěným textem, na str. 91.)

V jazyce HTML se lze orientovat na základě znaků <, >, \ a uvozovek (", '). Řešením je tedy návrhový vzor stavový automat, iterace textem a příslušná reakce na tyto znaky (viz obrázek 11.6). Tato reakce zahrnuje změnu stavu a případné přidání úseku od poslední značky do aktuální pozice do pole textových úryvků.

```
127
128     public generateTextParts(text: string): [textPart: string, offset: number][] {
129         while (this.index < text.length) {
130             this.state.follow(this, text)
131         }
132         return this.textParts
133     }
134 }
```

Figure 11.6: Cyklus extrakce textových úryvků

Obrázek 11.7 ukazuje přechod z textového úryvku do komentáře nacházejícího se uvnitř textového úryvku. Ostatní přechody jsou podobné.

```
4
5  class Correction implements TextMeaning {
6
7      private startIndex: number
8
9      constructor(context: Context) {
10         this.startIndex = context.getIndex()
11     }
12
13     public follow(context: Context, text: string): void {
14
15         if (text.substring(context.getIndex(), context.getIndex() + 4) == "<!--") {
16             context.addTextPart([text.substring(this.startIndex, context.getIndex()), this.startIndex])
17             context.setState(new CommentInCorrection())
18             context.incrementIndex(4)
19             return
20         }
```

Figure 11.7: Detekce komentáře a změna stavu

Na tomto obrázku je vidět i hlavička třídy Correction – stavu, kdy se iteruje v textovém úryvku. Implementuje rozhraní stavu nazvané TextMeaning, podobně jako tyto třídy:

- Tag – uvnitř textu značky,
- Attribute – atribut uvnitř značky,
- CommentInTag – komentář uvnitř značky,
- CommentInCorrection komentář uvnitř textového úryvku

Pro extrakci textových částí nebylo možné znovu použít DOMParser (stejně jako při validaci), jenž by sestavil objekt dokumentu, který by poté mohl vracet informace o různých značkách. Dokument si totiž DOMParser mírně upravuje. Například do tabulek přidává značku <tbody>a </tbody>, pokud v původním textu chybí. Důsledkem toho by se zničilo indexování změn v dokumentu.

11.5 Regulární výrazy

Aplikace jednotlivých pravidel pro vkládání NM je řešena různými regulární výrazy v komponentě Regular Expressions. Pomocí nich jsou vyhledávány výrazy, které spadají pod určitá typografická pravidla (více o typografických pravidlech viz kapitola 2, Pravidla pro vkládání, str. 5). Předmětem této kapitoly je konkrétní podoba regulárních výrazů a jejich výběr podle aktuální konfigurace.

11.5.1 Soubor regularExpressionsDB.ts

Pravidla konfigurovaná uživatelem jsou uložena v souboru settings.json, ovšem nejsnazším způsobem, jak databázi s ostatními regulárními výrazy exportovat pro zbytek aplikace, je psát ji v jazyce TypeScript. Data s regulárními výrazy jsou tedy dostupná v souboru regularExpressionsDB.ts. Tudíž odpadá nutnost jakéhokoli převodu z řetězce na regulární výraz a v případě této aplikace to nevýhody má pouze nepatrné.

Implementace hrubých pravidel z kapitoly 2 (na str. 10), je vidět na obrázku 11.8.

```
17
18  // rough and switchable
19  export const NBSP_AFTER_MATH: RegExp[] = [/\d [=<>≤≥+±×x·÷]/g]
20  export const NBSP_BEFORE_MATH: RegExp[] = [/[=<>≤≥+±×x·÷] \d/g]
21  export const NBSP_BEFORE_MINUSES: RegExp[] = [
22      /[-] [^\d]/g,
23      /[−] [^\d]/g]
24  export const NBSP_BEFORE_SLASHES: RegExp[] = [/\/ \b/g]
25
26  export const IV_NOT_NUMERALS: RegExp[] = [/[IV] [\wA-ZĚŠČŘŽÝÁÍÉÚŮa-zěščřžýáíéúů\d]+/g]
27
```

Figure 11.8: Regulární výrazy pro hrubá pravidla

11.5.2 Soubor regularExpressionsPicker.ts

Před čtením hodnot z DB je potřeba načíst uživatelskou konfiguraci, aby byla čtena jen potřebná data, v souboru regularExpressionsPicker.ts komponenty Regular Expressions Picker. Např. první dvě pole regulárních výrazů z ukázky na obrázku 11.8, NBSP_AFTER_MATH a NBSP_BEFORE_MATH, nikdy nejsou používána současně (vždy je použito právě jedno z nich). K tomuto vztahu a podobným situacím dochází mezi hrubými pravidly několikrát. Vybráno je vždy to pole regulárních výrazů, které odpovídá hodnotě přečtené z konfigurace (viz obrázek 11.9). O čtení konfigurace se však stará jiná komponenta, Settings.

```
20
21  function addRegexpsAccordingToConfiguration(regexps: RegExp[], regexpsConfiguration: ConfigData) {
22      if (regexpsConfiguration.wrapAroundMath === "před") {       //zalamuje se před MO, NM vkládána za MO
23          regexps.push(...regexpsDB.NBSP_AFTER_MATH)
24          if (!regexpsConfiguration.dashes) regexps.push(...regexpsDB.NBSP_BEFORE_MINUSES)
25          if (!regexpsConfiguration.slashes) regexps.push(...regexpsDB.NBSP_BEFORE_SLASHES)
26      }
27      else {      //zalamuje se za MO, NM vkládána před MO
28          regexps.push(...regexpsDB.NBSP_BEFORE_MATH)
29          regexps.push(...regexpsDB.NBSP_BEFORE_MINUSES)
30          regexps.push(...regexpsDB.NBSP_BEFORE_SLASHES)
31      }
32
```

Figure 11.9: Výběr a čtení regulárních výrazů

11.6 Nastavení

Klíčovým souborem pro uživatelskou konfiguraci je manifest, soubor package.json. Prostřednictvím něho VS Code ví, jaká nastavení uživateli nabídnout. Uživatelské preference jsou pak aplikací čtené s využitím VSCodeAPI.

11.6.1 Soubor settingsAccess.ts

Tento soubor komponenty Settings obsahuje funkce ke čtení konfigurovaného nastavení a implicitní hodnoty pro případ, že by se je nepodařilo načíst (pokud by byl soubor package.json poškozen).

Definuje přitom typ ConfigData, který slouží pro přenos dat do operace pro výběr regulárních výrazů v komponentě Regular Expressions (data transfer object). Jak je vidět na obrázku 11.10, jde o množinu hodnot potřebných pro tento výběr.

```
10
11  export type ConfigData = {
12      wrapAroundMath: string,
13      dashes: boolean,
14      slashes: boolean,
15      romanCaution: boolean,
16      datesValidation: boolean,
17      monthYearSeparation: boolean,
18      wrapAfterDegrees: boolean,
19      wrapInMathParentheses: boolean,
20      custom: RegExp[]
21  }
22
```

Figure 11.10: Datový typ ConfigData

11.6.1.1 Čtení konfigurace

Konfigurační nastavení je čtena voláním:

```
configuration.get<<typ hodnoty>>('<identifikátor nastavení>')
```

Aby toto volání mohlo být úspěšné, v manifestu projektu package.json musí být nastavení s příslušným identifikátorem deklarováno, podobně jako to ukazuje obrázek 11.11.

```
51    "configuration": {
52      "title": "nbspcorrector",
53      "properties": {
54        "nbspcorrector.nonBreakingSpacesNotation": {
55          "order": 0,
56          "type": "string",
57          "enum": [
58            " ",
59            " ",
60            " ",
61            " "
62          ],
63          "default": " ",
64          "description": "Vyber zápis používaný pro nezlomitelné mezery."
65        },
```

Figure 11.11: Deklarace nastavení

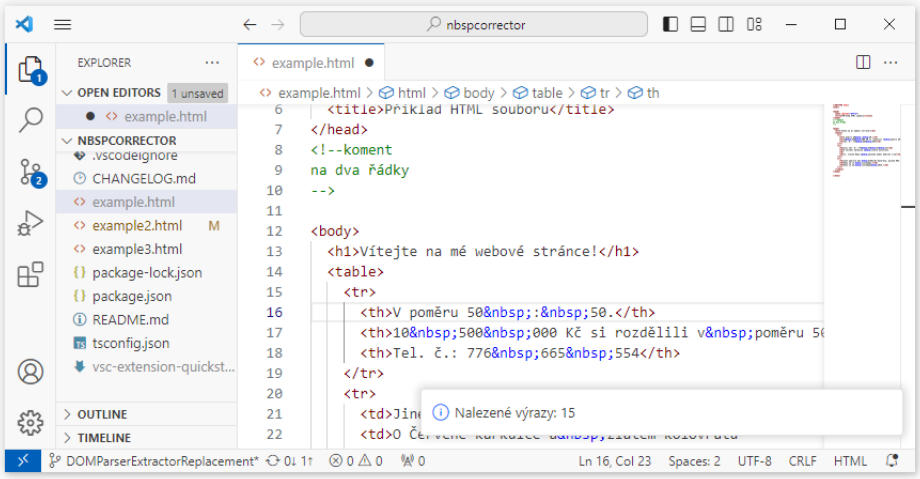

Figure 11.12: Upozornění po skončení vkládání NM

```
1  import * as vscode from 'vscode';
2
3  export function alarm(text: string) {
4      vscode.window.showErrorMessage(text)
5  }
6
7  export function inform(text: string) {
8      vscode.window.showInformationMessage(text)
9  }
```

Figure 11.13: Enter Caption

11.7 Zprávy uživateli

Uživatel je o výsledku operace vkládání NM informován způsobem přímo závislým na prostředí aplikace. V tomto případě byla zvolena rozhraním VSCodeAPI nabízená funkcionalita upozornění, která se objevují v pravém spodním rohu okna.[54] Ilustruje to obrázek 11.12, kde je vidět umístění upozornění – vpravo dole.

11.7.1 Soubor messager.ts

Jde pouze o primitivní adaptér, který volá metody na rozhraní VSCodeAPI (viz obrázek 11.13). To rozlišuje zprávy chybové (obrázek 11.14) a informační (obrázek 11.15), které se od sebe liší barevnou ikonou.

Figure 11.14: Enter Caption

Figure 11.15: Enter Caption

11.7.2 Závěr implementace

Byly implementována všechna potřebná funkcionalita a téměř všechny systémové požadavky (dostupné v kapitole 9 na str. 49). Seznam implicitních vyhledávacích výrazů implementován nebyl, jelikož jako nejvhodnější prostředek k publikaci těchto neměnných informací byl vyhodnocen soubor readme.md.

Bohužel konfigurace vlastních regulárních výrazů spočívá v psaní těchto výrazů v souboru settings.json, což není příliš uživatelsky přívětivé řešení. V ideálním případě by tyto regulární výrazy byly zapisovány a mazány přímo v rozhraní nastavení rozšíření. Bohužel se však nepodařilo najít způsob, jak toho dosáhnout. Implementaci lepšího konfiguračního rozhraní lze plánovat do budoucna, avšak je třeba poznamenat, že jsme omezeni tím, co v rozhraní nastavení umožní VS Code.

Kapitola 12

Testování

Tato kapitola patří testování, důležité části vývoje SW, bez níž by aplikace pro vkládání NM jistě nefungovala. V následujících odstavcích je popisováno, jakými technikami byly testovány které části aplikace, jaké chyby jimi byly odhaleny, jaké byly jejich dopady a jak byly opraveny.

12.1 Technologie

Testy lze spouštět pomocí nástroje vývojářů VS Code v příkazovém řádku. Toto rozhraní využívá testovacího frameworku Mocha.[62] Ten běží v prostředí Node.js. Testy framework spouští sériově.[63] Příklad jeho použití v kódu této aplikace (volání funkce suite) lze vidět na obrázku 12.1.

```
6
7   suite('CorrectionExecution test suite', () => {
8       const nbspNotation = settingsAccess.loadNBSPNotation()
9       const replacementGetter = (replaced: string) =>
10          replaced.replace(/ /g, nbspNotation)
11
12  >   test('createRegexpCorrections test 1', () => {···
20      })
21  >   test('createRegexpCorrections test 2', () => {···
28      })
29      //...})
30
```

Figure 12.1: Enter Caption

12.2 Testy

Testování bylo zaměřeno na složitější komponenty systému, kterými jsou Document Access, Spaces Corrector a Regular Expressions. Byly přitom použity unit a end-to-end testy. Na obrázcích níže jsou testované komponenty zvýrazněné červenou barvou.

12.2.1 Unit testy

Unit testy byla testována komponenta Document Access. Důvodem byla především práce s do té doby neznámým rozhraním VSCodeAPI pro čtení a úpravu dokumentů. Ačkoli jde o primitivní testy, byly důležité, protože DocumentAccess (viz diagram na obrázku 12.2) je první místo styku programu s dokumentem, a tudíž se komponenta také stávala jako první podezřelou, když aplikace nefungovala.

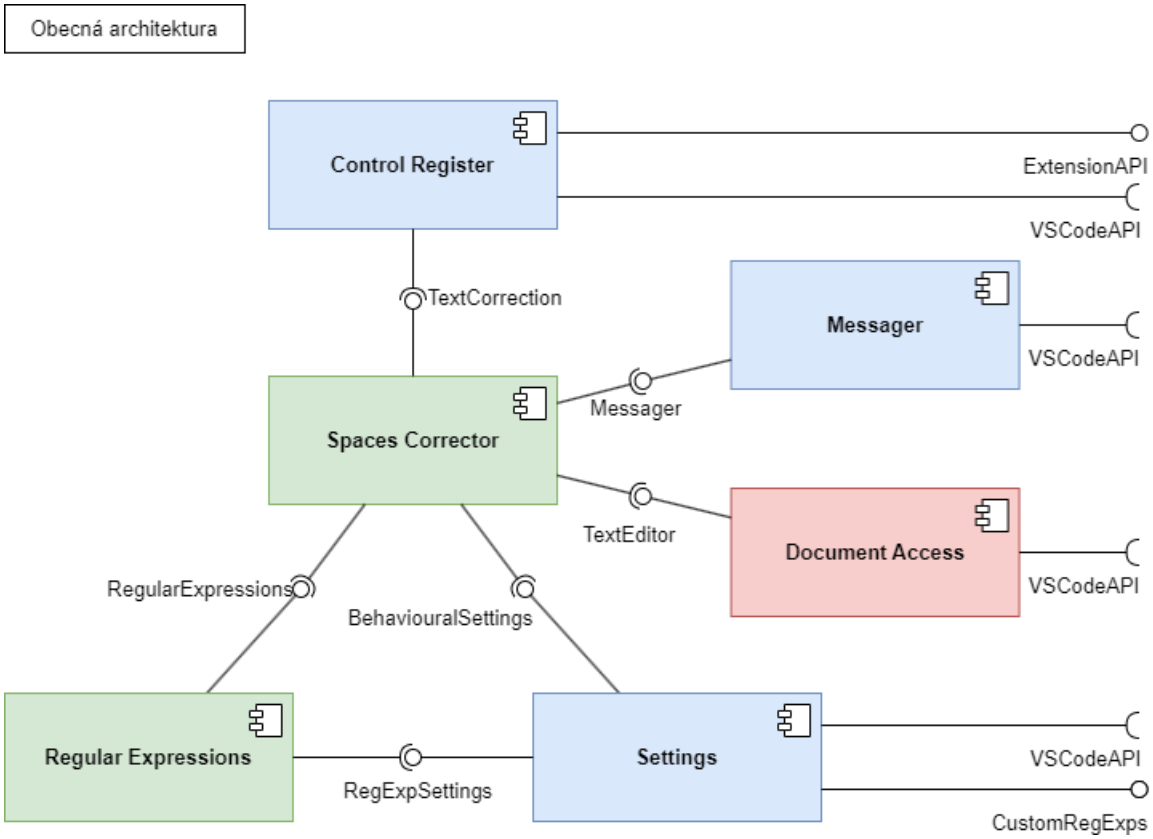

Figure 12.2: Kontext testované komponenty Document Access

Dále byla pomocí unit testů ověřována správnost regulárních výrazů v databázi (viz diagram na obrázku 12.3). Regulární výrazy byly psány ručně a i sebemenší chyba v některém z nich mohla způsobit, že příslušné typografické pravidlo nebude aplikováno a že se objeví false negatives.

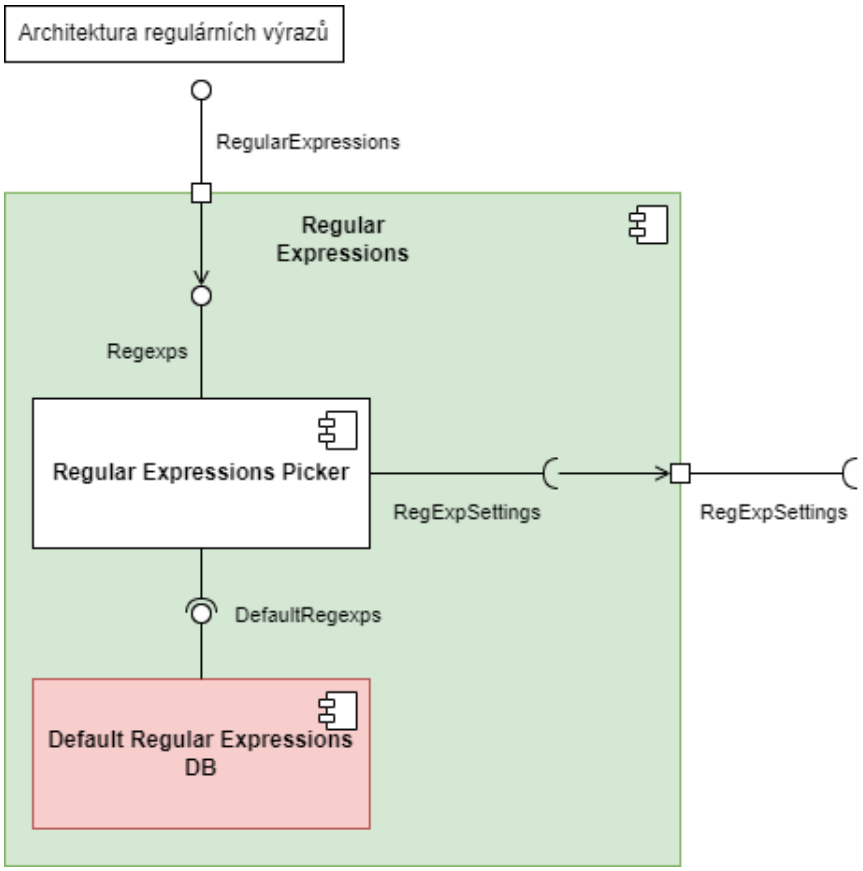

Figure 12.3: Kontext testované komponenty Regular Expressions DB

Nakonec unit testy sloužily k testování generování změn pro textové úryvky obsahující nějaký jev, kde mají být vkládány NM. To měla na starosti komponenta Correction Creator (viz diagram na obrázku 12.5). Ukázka jednoho z těchto testů je k vidění na obrázku 12.4.

12.2.2 End-to-end testy

Tyto testy byly nutné, jelikož cílem byla plně fungující aplikace. Aplikace byla ručně vyzkoušena na několika příkladech HTML souborů. Pozornost byla věnována tomu,

- zda aplikace nalezne všechna místa pro vložení NM
- zda najde místa pro přepsání NM na klasické mezery,
- zda všechny opravy správně zapíše do dokumentu,
- zda celý proces nebude trvat nepřiměřeně dlouhou dobu.

```
11
12      test('createRegexpCorrections test 1', () => {
13          const correctedString: string = "O výši odměny se právě jedná."
14          const regexp: RegExp = /(?<=^| )[O] \b/g
15          const createdCorrection = createCorrectionFromMatch(
16              regexp.exec(correctedString)!, regexp, replacementGetter)
17          assert.equal(createdCorrection[0], "O" + nbspNotation)
18          assert.equal(createdCorrection[1], 0)
19          assert.equal(createdCorrection[2], 2)
20      })
21
```

Figure 12.4: Ukázka testu generování oprav

Soubory, na kterých byla aplikace testována, jsou krátké soubory HTML vytvořené pouze pro účely testování i soubor stránky Zpovědnice [1] upravený pro účely testování správnosti spočítaných indexů mezer v textu dokumentu.

[1] https://www.zpovednice.eu/

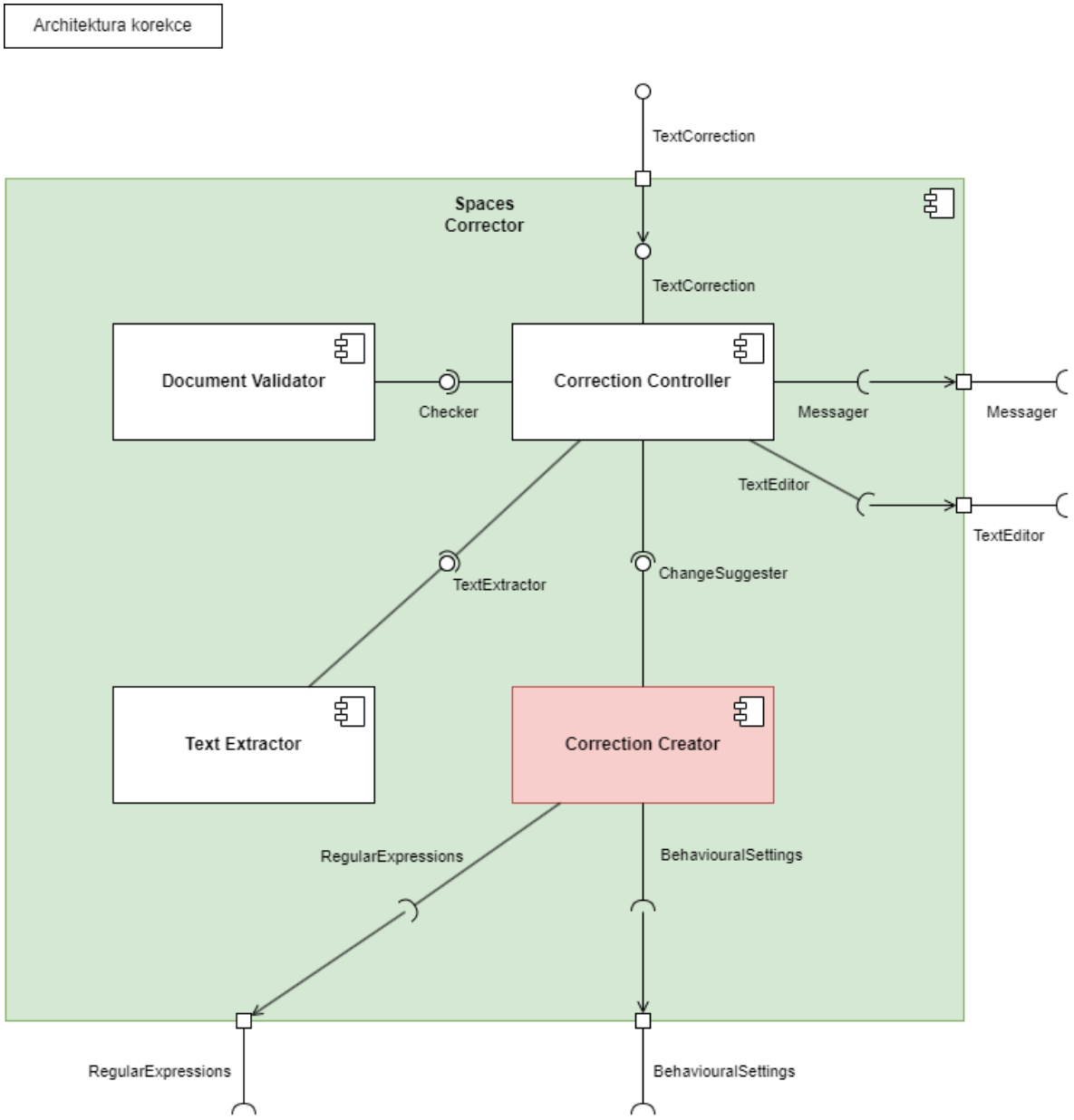

Figure 12.5: Kontext testované komponenty Correction Creator

12.3 Odhalené chyby a opravy

Chyby, které se podařilo odhalit, jsou většinou závažného charakteru. Většinu se však podařilo také opravit.

12.3.1 Regulární výrazy

Mezi regulárními výrazy se pomocí unit testů objevily chyby, a tak některá typografická pravidla v textech nebyla aplikována. Ačkoli se někdy jednalo o nadměrně dlouhé regulární výrazy, podařilo se všechna místa s chybnými znaky v regulárních výrazech identifikovat a tyto chyby opravit.

12.3.2 Obnova výchozího stavu

Pomocí end-to-end testů bylo objeveno, že prvního, druhé a každé další vkládání NM má odlišné výsledky. Důvodem byla chybějící inicializace množiny oprav na původní hodnotu po každém vkládání. Data k opravám zůstávala v paměti i po jejich provedení. Odstranění této chyby bylo ověřeno opětovným testováním.

12.3.3 Vypočítávané indexy mezer

End-to-end testy prokázaly také chybné výpočty indexů, které se projevovaly např. vkládáním NM na nesprávných místech, ačkoli počet vložených NM byl správný. Pomocí rozhraní DOMParser lze sice snadno ověřit, zda se jedná o validní dokument, ukázalo se však, že s bílými znaky ve vstupním textu pracuje pro účel aplikace zcela nevhodným způsobem (přidává je nebo je odebírá), a tak nakonec pracuje s jiným textem, než je ve skutečném dokumentu. Tudíž se nabízelo indexy přepočítávat podle počtu bílých znaků nacházejících se před místem oprav mezer.

Po vyrovnávání indexů prostřednictvím počítání bílých znaků testy opět neprocházely. Nástroj DOMParser totiž do dokumentu některé přidává některé značky, pokud chybí. Jelikož se nepodařilo najít, které všechny značky DOMParser přidává, upravuje či odebírá, nelze ho použít pro vybírání textu z HTML dokumentu.

12.3.4 Stávající chyba indexů

Implementace vlastního parseru (viz kapitola 12, Implementace, na str. 68) způsobila, že některými testy zabývající se problémem výpočtů indexů začal program procházet. Nicméně nové testy ukázaly, že při zpracování některých HTML souborů indexy stále nepočítá správně. Příčina těchto chyb nebyla dosud přesně lokalizována, avšak testování naznačuje, že se mohou objevit jen, pokud je v dokumentu více určitých bílých znaků (odřádkování), za nimiž následuje text, v němž má být provedena úprava. Takový chybný výsledek může vypadat například jako na obrázku 12.6, kdy je s každým dalším řádkem mezera posunuta. (Původní řetězec byl "Mapa je v měřítku 1 : 50 000.") Uživatelé jsou

tedy nuceni před použitím aplikace tato odřádkování odstranit, což však naštěstí většinou nemá vliv na výsledné zobrazení webové stránky, jelikož tato odřádkování prohlížeč obvykle nahrazuje mezerami.[64]

```
32 <body bgcolor="#000000" text="#FFFFFF" link="#FFFFFF" vlink="#C0C0C0" alink="#FFFFFF">
33     Mapa je v měřítku 1 : 50 000.
34     Mapa jv v měřítku : :  0 000.
35     Mapav e v měřít : 1   50 000.
36     Mav  je v měř : u   : 50 000.
37
```

Figure 12.6: Důsledek chybně vypočtených indexů

12.3.5 Závěr testování

Unit testy byla testována především základní funkcionalita, s pomocí end-to-end testů částečně také funkčnost konfiguračního nastavení a uživatelská přívětivost. Unit testy se ukázaly jako efektivnější metoda, jak funkcionalitu ověřit a chyby opravit a předpokládá se také, že pokud by jich bylo vytvořeno více, čas potřebný na opravy by se patrně zkrátil. Nicméně celkovou použitelnost aplikace bylo možné posoudit až pomocí end-to-end testů.

Na základě výsledků testů byly prováděny úpravy, kterými se většinu chyb podařilo odstranit. (Odstranění bylo vždy ověřováno opětovným testováním.) V pozdější fázi vývoje byla aplikace testována na některých specificky psaných souborech HTML, kde se ukázalo, že jsou NM vkládány na nesprávná místa. Celkově vzato tedy testování ověřilo, že je produkt připraven k použití, i když stále existuje prostor pro zlepšení zejména v oblasti uživatelské přívětivosti.

Part III

Výsledky

Kapitola 13

Distribuce

Ze zadání vyplývá publikace zdrojového kódu na GitHubu. Uživatele a další vývojáře může tento kód zajímat z různých důvodů a navíc to aplikaci přidává na důvěryhodnosti. V následujících odstavcích jsou zvažovány způsoby, jak ji cílovým uživatelům přívětivou cestou distribuovat.

13.1 Způsoby distribuce

Možnosti, jak lze aplikaci distribuovat, se odvijí především od její podoby. V případě této bakalářské práce jde o rozšíření pro VS Code.

13.1.1 Visual Studio Code Marketplace

Nejen že nabídku VS Code Marketplace lze otevřít na stránkách VS[1], ale je možné se do ní snadno a rychle dostat také prostřednictvím VS Code jedním kliknutím. Všechna rozšíření, včetně produktu této práce pod názvem *Czech   Corrector*, jsou tak na jednom místě a vyhledávat lze mezi nimi podle názvu, kategorie anebo dalších vlastností. VS Code umožňuje snadnou a rychlou instalaci na místě, kde lze rozšíření rovnou vyzkoušet. [54]

13.1.2 Webové stránky

Podobně lze stažení aplikace nabízet na webových stránkách – vlastních, či na serverech typu Ulož.to nebo Slunečnice. Uživatelům by musel být pak téměř vždy dopraven odkaz na webovou stránku, což by oproti obchodu VS Code Marketplace zhoršovalo dostupnost a důvěryhodnost a postup instalace by byl náročnější. Pro tyto nevítané vlastnosti zůstal tento způsob distribuce nevyužit.

[1] https://marketplace.visualstudio.com/vscode

13.2 Rozhodnutí o distribuci

Aplikace byla distribuována v obchodě VS Code Marketplace pod názvem *Czech   Corrector*[2]. Jde o uživatelsky nejpřívětivější a zároveň nejsnazší cestu, jak uživatelům distribuovat rozšíření i jeho průběžné aktualizace. Více než pravděpodobně se také nejspíše jedná o první místo, kde uživatelé editoru VS Code hledají jeho rozšíření.

[2]https://marketplace.visualstudio.com/items?itemName=WardenSpirit.nbspcorrector&ssr=false

Kapitola 14

SWOT analýza aplikace

SWOT analýza prováděna po publikaci a několika opravách aplikace vyhodnocuje strategické vlastnosti projektu. Výstupem analýzy je např. hodnota úspěšnosti aplikace užitečná pro zhodnocení výsledků práce nebo v případě ambicí k monetizaci aplikace.

14.1 SWOT matice

U každého z prvků SWOT analýzy je vyhodnocena důležitost (podíl vlivu v rámci skupiny) a spokojenost (síla vlivu). Konkrétní hodnoty ukazují tabulky 14.1, 14.2, 14.3 a 14.4.

Popis	Důležitost	Spokojenost
Nástroj je snadno k nalezení, stažení a použití (distribuce přímo prostřednictvím obchodu s rozšířeními, vkládání bývá téměř okamžité).	0,3	4
Nástroj nemá konkurenci svého typu. (Viz kapitola 3, Existující řešení, na str. 17).	0,15	4
Nástroj aplikuje nejvíce typografických pravidel (v implicitním nastavení) mezi obdobnými programy.[1].	0,1	3
Návrh umožňuje snadnou migraci aplikace na jiné platformy. (Návrh viz kapitola 10 na str. 53.)	0,15	3
Návrh umožňuje snadnou migraci do jiných kontextů než HTML. (Též vychází z kapitoly Návrh.)	0,15	1
Návrh umožňuje snadno zapojit umělou inteligenci nebo zkrátka využít externí rozhraní, které by opravu textu provádělo lépe. (Opět viz kapitola Návrh.)	0,15	2

Table 14.1: Silné stránky

Hodnota silných stránek: $0{,}3 \times 4 + 0{,}15 \times 4 + 0{,}1 \times 3 + 0{,}15 \times 3 + 0{,}15 \times 1 + 0{,}15 \times 2 = \mathbf{3}$

Popis	Důležitost	Spokojenost
Aplikace při jistých způsobech odřádkování v dokumentu nefunguje správně.	0,2	3
Vkládání není dokonalé bez důkladné uživatelské konfigurace. (Viz kapitoly 2, Pravidla pro vkládání, na str. 5 a 8, Uživatelská konfigurace, na str. 37)	0,05	4
Aplikace je určena pouze pro uživatele nástroje VS Code.	0,2	1
Uživatelská konfigurace nemá uživatelsky přívětivé rozhraní.	0,2	4
Nástroj je určen pouze autorům česky psaných dokumentů a povědomí o nezlomitelných mezerách nemá každý, tudíž je počet potenciálních uživatelů omezen.	0,3	2
Vývoj aplikace závisí na jediném vývojáři.	0,05	5

Table 14.2: Slabé stránky

Hodnota slabých stránek:
$0{,}2 \times 3 + 0{,}05 \times 4 + 0{,}2 \times 1 + 0{,}2 \times 4 + 0{,}3 \times 2 + 0{,}05 \times 5 = \mathbf{2{,}65}$

Popis	Důležitost	Spokojenost
Počet instalací pravděpodobně poroste. (Zatím roste každým týdnem průměrně o 4.[65]. Pro srovnání s původními cíli viz kapitola 1, Úvod, na str. 1)	1	4

Table 14.3: Příležitosti

Hodnota příležitostí: $1 \times 4 = \mathbf{4}$

Popis	Důležitost	Spokojenost
Aplikace využívající umělé inteligence by pro vkládání nezlomitelných mezer mohly nacházet více míst, čehož by mohla využít nová konkurence.	0,5	1
Objeví se nová pravidla pro vkládání nezlomitelných mezer, na jejichž aplikaci systém regulárních výrazů používaný nyní již nestačí. Od roku 1987 nabylo platnosti celkem 6 vydání příslušné normy (ČSN 01 6910).[66]	0,2	1
Češtinu na internetových stránkách postupně nahradí angličtina. Přispívá k tomu fakt, že mezi lety 2014 a 2022 počet anglicky mluvících vzrostl o (původní) třetinu na více než 1,4 miliardy.[67]	0,3	3

Table 14.4: Hrozby

Hodnota hrozeb: 0,5 × 1 + 0,2 × 1 + 0,3 × 3 = **1,6**

14.2 Závěr

Podstatným výstupem je zde výstupní hodnota analýzy, která napovídá, jak je projekt úspěšný. Spočítá se jako součet silných stránek, příležitostí a opačných hodnot slabých stránek a hrozeb: 3 + 4 - 2,65 - 1,6 = **2,75**. Jelikož je výsledek, 2,75 jednoznačně kladný, lze aplikaci v rámci SWOT analýzy považovat za úspěšnou. Je třeba poznamenat, že k výpočtům vedla pouze jedna příležitost, avšak velmi podstatná.

Projekt se nyní nachází v tzv. *modrém oceánu*, tzn. zatím na trhu není produkt, který by mu v rámci VS Code konkuroval. Jeho cílem je šetřit čas svým uživatelům, a pokud opomineme, že se jedná o nedílnou součást bakalářské práce, míra úspěchu je jejich ušetřenému času přímo úměrná. Aplikace má tedy zatím nenaplněný potenciál v počtu uživatelů. Navíc hrozby se naplní s menší pravděpodobností, pokud se podaří aplikaci včas vylepšit.

Z toho důvodu by vhodnou strategií pro následující čtvrtletí mohly být strategie *max-max*, nebo *min-max*. První těží ze silných stránek, druhá se snaží redukovat ty slabé. Obě strategie mají ambice využít významnou příležitost, kterou představuje stále rostoucí počet instalací. Uživatele je však potřeba si také udržet.

Konkrétní kroky, které lze na základě této analýzy podniknout, aby produkt šetřil čas svým uživatelům, a plnil tak svůj cíl co nejlépe, zní:

1. oprava chyb, aby byli uživatelé ušetřeni zklamání a neodcházeli;

2. nalezení způsobu, jak vylepšit prezentační stránku aplikace, což by uživatelům ještě více usnadnilo rozhodování, zda aplikaci instalovat, a navíc jde o levný krok;

3. vývoj lepšího rozhraní pro uživatelskou konfiguraci, aby se dostala do povědomí jako seriózní;
4. rozšíření implicitně podporovaných pravidel.

Kapitola 15

Závěr

Závěr shrnuje výsledky této bakalářskou práci a porovnává je s vytyčenými cíli.

15.1 Zhodnocení výstupů bakalářské práce

Podařilo se vytvořit fungující aplikaci pro automatickou úpravu dokumentů v rámci zalamování řádků. Co do počtu implementovaných pravidel jich bylo v implicitním nastavení implementováno 16 z celkového počtu 20 (popisovaných v kapitole 2, Pravidla pro vkládání, na str. 5). Směrodatné informace však přinese až odezva uživatelů. Mezi nalezenými aplikacemi pro automatické vkládání nezlomitelných mezer do českého textu má tato aplikace nejširší pokrytí českých typografických pravidel.[1] V podobě rozšíření do nástroje VS Code ji mohou využívat autoři HTML dokumentů používající tento nástroj pro efektivní vkládání nezlomitelných mezer, a ušetřit tak svůj čas.

V této bakalářské práci jsou popisovány jednotlivé kroky od vytyčených cílů a získávání informací přes návrh, implementaci a testování po zhodnocení výsledků. V obchodě VS Code Marketplace, kde je publikována, má nakonec 52 dnů od publikace první verze, 37 instalací, což je po odečtení počtu tří vlastních instalací přibližně 227 % prvního cíle této bakalářské práce (cíle viz kapitola 1, Úvod, na str. 1). Aplikaci prezentuje text s odkazem na typografická pravidla, čímž je plněn také druhý cíl.

SWOT analýza provedená měsíc a půl po uvedení aplikace na veřejnost v obchodě VS Code Marketplace o aplikaci navíc říká, že hodnoty jejích silných stránek a příležitosti převažují nad slabými stránkami a hrozbami. To je známkou jejího úspěchu a zároveň důvodem pro optimistický pohled na její budoucí vývoj.

[1] Porovnávána byla s aplikacemi   replacer, Vlna a Seznam Médium viz kapitola 3, Existující řešení, na str. 17.

15.2 Potenciál aplikace v případě dalšího vývoje

Jak bylo plánováno během návrhu architektury, větší část kódu je psána nezávisle na prostředí nástroje VS Code a rozhraní pro jeho rozšíření, tudíž se nepředpokládá, že by bylo složité využít kód také v jiných prostředích. Předně se pro to nabízí obdobné nástroje společnosti JetBrains nebo webové prohlížeče.

15.3 Získané zkušenosti

Před vypracováním této bakalářské práce nebyly mé znalosti typografických českého jazyka pro psaní nezlomitelných mezer a zkušenosti s psaním rozšíření do textového editoru VS Code téměř žádné. Během rešerše jsem pracoval s technickými normami ČSN 01 6910, které jsou v oblasti zalamování řádků formálním a obecně uznávaným souborem těchto pravidel, abych tyto znalosti následně využil pro vytvoření nástroje, který tato pravidla v co možná největší míře automaticky opravuje, a ušetřil tak čas jeho uživatelům. Mezitím jsem se všechna tato pravidla naučil, což jsem využil mj. i při psaní této bakalářské práce, a získal jsem základní zkušenosti s psaním rozšíření pro nástroj VS Code.

Zkratky

Písmo a typografie

NM	Nezlomitelná mezera
ASCII	American Standard Code for Information Interchange
ČSN	české technické normy

Značkovací jazyk

HTML	Hypertext Markup Language
XHTML	Extended Hypertext Markup Language
XML	Extensible Markup Language
DOM	Document Object Model

Ostatní zkratky

API	Aplikační programové rozhraní (Application programming interface)
SW	Software
IDE	Integrated Development Environment
RV	regulární výraz
RVP	rámcový vzdělávací program
MS	Microsoft
VS	Visual Studio
SRQ	Systémový požadavek (System requirement)

Seznam příloh

Soubory

nbspcorrector.zip – soubor komprimovaných zdrojových kódů a dalších součástí aplikace

Czech_ _Corrector.VSIX – soubor rozšíření; instalaci lze provést ve VS Code pomocí volby *Install from VSIX...* v menu rozšíření.

Bibliografie

[1] Ministerstvo školství, mládeže a tělovýchovy. RVP ZV - Rámcový vzdělávací program pro
základní vzdělávání. Dostupné z: https://www.edu.cz/rvp-ramcove-vzdelavaci-programy/ramcovy-vzdelavacici-program-pro-zakladni-vzdelavani-rvp-zv/

[2] Balada, Jan a další. Rámcový vzdělávací program pro gymnázia. Dostupné z: https://www.edu.cz/wp-content/uploads/2020/08/RVPG-2007-07_final.pdf

[3] Centrum pro zjišťování výsledků vzdělávání. Katalog požadavků zkoušek společné části maturitní zkoušky. Dostupné z: https://maturita.cermat.cz/files/files/katalog-pozadavku/katalog-pozadavku-2018-CJL.pdf

[4] Centrum pro zjišťování výsledků vzdělávání. Specifikace požadavků pro jednotnou přijímací zkoušku v přijímacím řízení na střední školy v oborech vzdělání s maturitní zkouškou. Dostupné z: https://prijimacky.cermat.cz/files/files/dokumenty/specifikace-pozadavku/Specifikace_2022-2023/CJLSPECIFIKACEPOZADAVKU2022.pdf

[5] Centrum pro zjišťování výsledků vzdělávání. Jednotná přijímací zkouška 2023. Dostupné z: https://prijimacky.cermat.cz/menu/jednotna-prijimaci-zkouska

[6] Události historie. Dostupné z: http://udalosti-historie.comehere.cz/index.php?article=9

[7] Ústav pro jazyk český AV ČR, v. v. i. ČSN 01 6910 (2014) Úprava dokumentů zpracovaných textovými procesory. Dostupné z: https://ujc.avcr.cz/expertni-cinnost/csn016910/faq.html

[8] Ústav pro jazyk český AV ČR, v. v. i. Zalomení řádků a nevhodné výrazy na jejich konci. Dostupné z: https://prirucka.ujc.cas.cz/?id=880&dotaz=d%C4%9Blen%C3%AD%20slov%20na%20konci%20%C5%99%C3%A1dku

[9] Ústav pro jazyk český AV ČR, v. v. i. ČSN 01 6910 (2014) – Úprava dokumentů zpracovaných textovými procesory. Dostupné z: https://ujc.avcr.cz/expertni-cinnost/csn016910

[10] Ústav pro jazyk český AV ČR, v. v. i. Římské číslice. Dostupné z: https://prirucka.ujc.cas.cz/?id=793

[11] Ústav pro jazyk český AV ČR, v. v. i. Komunikace s Jazykovou poradnou. "Telefonický rozhovor dne 13. července 2023".

[12] Beroušek, J. Novinky a historie verzí. Dostupné z: http://www.nedivse.cz/doplnovani-pevnych-mezer/novinky-historie-verzi.php

[13] Beroušek, J. Automatické vkládání pevných mezer. Dostupné z: http://www.nedivse.cz/doplnovani-pevnych-mezer/index.php

[14] Beroušek, J. Nápověda. Dostupné z: http://www.nedivse.cz/doplnovani-pevnych-mezer/napoveda.php

[15] Žára, O. Konzultace s vedoucím práce, RNDr. Ondřejem Žárou. Konzultace v letním semestru 2022/2023 a zimním semestru 2023/2024.

[16] Olšák, P. Index of /ftp/olsak/vlna. "soubor vlna.txt", n.d. Dostupné z: https://petr.olsak.net/ftp/olsak/vlna/

[17] Olšák, P. Index of /ftp/olsak/vlna. "soubor vlna.w". Dostupné z: https://petr.olsak.net/ftp/olsak/vlna/

[18] Seznam.cz a.s. Seznam Médium. Dostupné z: https://admin.medium.seznam.cz/

[19] W3C. W3C Markup Validation Service. Dostupné z: https://validator.w3.org/

[20] Practical Typography. Prevent Awkward Breaks. Dostupné z: https://practicaltypography.com/nonbreaking-spaces.html

[21] Mozilla. lang. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/lang

[22] Vondráková, N. Nejnavštěvovanější webové stránky 2023. Dostupné z: https://sitevhrsti.cz/nejnavstevovanejsi-webove-stranky/

[23] Google. Google. Dostupné z: https://www.google.com/

[24] Seznam.cz a.s. Seznam.cz. Dostupné z: https://www.seznam.cz/

[25] YouTube, LLC. YouTube. Dostupné z: https://www.youtube.com/

[26] Facebook. Facebook. Dostupné z: https://www.facebook.com/

[27] Novinky.cz. Novinky.cz. Dostupné z: https://www.novinky.cz/

[28] iDNES.cz. iDNES.cz. Dostupné z: https://www.idnes.cz/

[29] Sport.cz. SPORT.CZ. Dostupné z: https://www.sport.cz/

[30] Super.cz. Super.cz. Dostupné z: https://www.super.cz/

[31] Seznam.cz a.s. stream. Dostupné z: https://www.stream.cz/

[32] Wikipedia. Wikipedia. Dostupné z: https://www.wikipedia.org/

[33] Routley, N. Visual Capitalist - Top 50 Most Visited Websites. Dostupné z: https://www.visualcapitalist.com/top-50-most-visited-websites/

[34] X Corp. X. Dostupné z: https://twitter.com/

[35] Instagram. Instagram. Dostupné z: https://www.instagram.com/

[36] Baidu. Baidu. Dostupné z: https://www.baidu.com/

[37] Dzen. Dzen. Dostupné z: https://dzen.ru/?yredirect=true

[38] Yahoo. yahoo! Dostupné z: https://www.yahoo.com/

[39] WhatsApp Inc. WhatsApp. Dostupné z: https://www.whatsapp.com/

[40] Mozilla. xml:lang. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/xml:lang

[41] Mozilla. HTML elements reference - HTML: HyperText Markup Language. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Language

[42] ředitelství pro komunikaci, G. Evropská unie. Dostupné z: https://european-union.europa.eu/index_cs

[43] Norwegian Meteorological Institute and the Norwegian Broadcasting Corporation. YR. Dostupné z: https://www.yr.no/en

[44] Mozilla. ¡span¿: The Content Span element. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/span

[45] Adobe. Panel Glyfy – přehled. Dostupné z: https://helpx.adobe.com/cz/indesign/using/glyphs-special-characters.html

[46] Mozilla. Entity. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/Entity

[47] iTecNote. HTML entitites - alternative to . Dostupné z: https://itecnote.com/tecnote/html-entitites-alternative-to-nbsp/

[48] WHATWG. HTML Living Standard. Dostupné z: https://html.spec.whatwg.org/multipage/named-characters.html#named-character-references

[49] JetBrains. JetBrains Marketplace. Dostupné z: https://plugins.jetbrains.com/

[50] Microsoft. Visual Studio Marketplace. Dostupné z: https://marketplace.visualstudio.com/vscode

[51] JetBrains. The Drive To Develop. Dostupné z: https://www.jetbrains.com/company/

[52] Taft, D. K. Microsoft VS Code: Winning developer mindshare. Dostupné z: https://www.techtarget.com/searchsoftwarequality/news/252496429/Microsoft-VS-Code-Winning-developer-mindshare

[53] Tung, Liam. Visual Studio Code: How Microsoft's plan is paying off. Dostupné z: https://www.zdnet.com/article/visual-studio-code-how-microsofts-any-os-any-programming-language-any-software-plan-is-paying-off/

[54] Microsoft. Visual Studio Code. Dostupné z: https://code.visualstudio.com/

[55] Microsoft. Your First Extension. Dostupné z: https://code.visualstudio.com/api/get-started/your-first-extension

[56] Mozilla. MVC. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/MVC

[57] Pearce, Richard. Pipeline Architecture. Dostupné z: https://www.cs.sjsu.edu/~pearce/modules/patterns/distArch/pipeline.htm

[58] Microsoft. Extension Anatomy. Dostupné z: https://code.visualstudio.com/api/get-started/extension-anatomy

[59] Microsoft. Contribution Points. Dostupné z: https://code.visualstudio.com/api/references/contribution-points

[60] FEL ČVUT. Události. Dostupné z: https://cw.fel.cvut.cz/old/_media/courses/bd6b36pjv/programovani_v_jave/07_udalosti.pdf

[61] Mozilla. async function. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function

[62] Microsoft. Testing Extensions. Dostupné z: https://code.visualstudio.com/api/working-with-extensions/testing-extension

[63] OpenJS Foundation. Mocha – simple, flexible, fun. Dostupné z: https://mochajs.org/

[64] Mozilla. How whitespace is handled by HTML, CSS, and in the DOM. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Whitespace

[65] WardenSpirit. Czech   Corrector. Dostupné z: https://marketplace.visualstudio.com/items?itemName=WardenSpirit.nbspcorrector&ssr=false#overview

[66] Česká agentura pro standardizaci. ČSN Online pro jednotlivce. Dostupné z: https://csnonline.agentura-cas.cz/Vysledky.aspx

[67] Statistics and data. The most Spoken Languages in the World - 1900/2022. Dostupné z: https://www.youtube.com/watch?v=6KBCzNt9MrM&ab_channel=Statisticsanddata