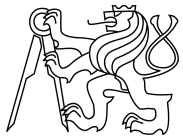
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Aplikace pro podporu psaní vědeckých článků
Student:	Bc. David Vondraš
Vedoucí:	doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je návrh a implementace softwarové aplikace, která bude napomáhat uživateli sestavit vědecký článek na základě předpřipravených textových bloků a kontextu obsahu sestavovaného textu.

Požadavky na aplikaci jsou:
- výběr předpřipravených částí textu pro jednotlivé části článku,
- modifikace vybraných částí textu,
- nabídka alternativ k formulacím vět s možností doporučování,
- jednoduchá interaktivní a iterativní práce,
- jádro aplikace bude realizováno pomocí RESTových služeb, aby bylo možné snadno vyměnit některé části (například doporučování formulací),
- výstupem aplikace bude návrh textu článku ve formátu LaTex nebo MSWord.

Postupujte v těchto krocích:
1. Zpracujte rešerši podobných přístupů.
2. Proveďte analýzu a návrh vlastního systému.
3. Zvolte vhodné implementační prostředky a systém implementujte.
4. Systém zdokumentujte a vhodným způsobem otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 15. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Aplikace pro podporu psaní vědeckých článků

Bc. David Vondraš

Vedoucí práce: doc. RNDr. Ing. Marcel Jiřina, Ph.D.

1. května 2017

Poděkování

Rád bych poděkoval vedoucímu práce doc. RNDr. Ing. Marcelu Jiřinovi, Ph.D. za poskytnuté rady při psaní této diplomové práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 1. května 2017 .....

Odkaz na tuto práci

Vondraš, David. *Aplikace pro podporu psaní vědeckých článků.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Cílem této diplomové práce je navržení webové aplikace, která bude uživatelům usnadňovat vytváření vědeckých článků nabízením předpřipravených vět a textových bloků v závislosti na zvoleném tématu. Aplikace obsahuje funkce potřebné pro našeptávání vět a pro vytvoření kompletního článku včetně vkládání příloh a generování citací. Práce obsahuje popis návrhu, implementace a testování aplikace.

Klíčová slova vědecký článek, automatické dokončování vět, našeptávání textu, kategorizace textu, struktura článku, téma článku, klíčová slova článku

Abstract

The aim of this diploma thesis is to design a web application that will facilitate the creation of scientific articles by offering pre-prepared sentences and text blocks depending on the chosen topic. The application contains the features needed for whispering sentences and for creating a complete article, including inserting attachments and generating quotations. The work includes the description of application design, implementation and testing.

Keywords scientific article, automatic sentence finishing, text whispering, text categorization, article structure, article topic, article keywords

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Vědecký článek je druh akademické publikace, která je obvykle vydávaná ve vědeckém časopise. Před uvedením článku je přezkoumáván odborníky na dané téma a často trvá velmi dlouho, než je zvalidován a uveřejněn. Aby měl článek šanci na schválení, musí autor při jeho vytváření dodržovat určitou strukturu a pravidla, která jsou pro vědecký článek stanovena. Autor tak musí kromě náročného výzkumu, který provádí, věnovat svůj čas a energii psaní rozsáhlého textu.

Autorům by jistě ušetřilo mnoho práce, kdyby mohli využít aplikaci, která by dodržovala všechna pravidla požadovaná pro strukturu vědeckého článku a k tomu jim nabízela fráze a části textu, které by mohli v článku využít a připsat k nim jen svoji myšlenku a detaily o svém výzkumu.

Ve své diplomové práci se budu věnovat možnostem, jak takovou aplikaci realizovat. Pokusím se navrhnout postupy potřebné pro nabízení textu podle tématu článku a naimplementuji zkušební aplikaci, do které zabuduji co nejvíce funkcí, které by efektivně napomáhaly při tvorbě vědeckých článků.

Cíl práce

Cílem této práce je ověřit možnost navrhnout a implementovat aplikaci, která by uživatelům co nejvíce ulehčovala psaní vědeckých článků nabízením předpřipravených textových bloků a kontextu sestavovaného článku. Cílem práce naopak není, aby výsledná aplikace tvořila články zcela bez zásahu uživatele. Při plnění tohoto cíle se budu držet hlavních bodů zadání práce, které jsou následující:

1. Aplikace bude svým uživatelům nabízet k výběru předpřipravené části textu specifické pro jednotlivé části článku.
2. Uživatelé budou mít možnost vybraný text modifikovat.
3. Uživatelům budou nabízeny a doporučovány alternativní formulace vět.

4. Aplikace bude umožňovat jednoduchou interaktivní a iterativní práci.
5. Aplikace bude realizována pomocí RESTových služeb, aby bylo možné snadno přidávat a nahrazovat jednotlivé části aplikace.
6. Uživatelé budou mít možnost si svůj vytvořený článek uložit do formátů MSWord a LaTex.

Struktura práce

Tato diplomová práce je rozčleněna na čtyři hlavní části a to: Analýza stávajících řešení a požadavky na aplikaci, Návrh, Realizace a Testování.

První část - **Analýza stávajících řešení a požadavky na aplikaci** - představuje požadovanou strukturu vědeckého článku, která musí být výslednou aplikací dodržována. Dále se věnuje analýze stávajících řešení a stanovení požadavků na aplikaci.

Následující kapitola s názvem **Návrh** se zabývá představením návrhu samotného systému. Obsahuje popis požadovaných funkcionalit a architektury aplikace. Je v ní také popsán návrh uživatelského prostředí a jednotlivé technologie využité při návrhu a implementaci.

Část práce nazvaná **Realizace** popisuje detaily týkající se implemetace klíčových částí systému. Jsou zde umístěny konkrétní příklady řešení důležitých funkcí. V této kapitole jsou také shrnuty možnosti budoucího vývoje výsledné aplikace.

Poslední kapitola - **Testování** - se detailně věnuje otestování aplikace. Obsahuje popis všech provedených testů od nejnižší úrovně (jednotkové testování kódu) až po testování reálnými uživateli. V závěru této části práce je rozebraný jeden ze článků, které vznikly během testování uživateli.

Kapitola 1

Analýza stávajících řešení a požadavky na aplikaci

Na začátku vývoje každého softwarového projektu je důležité provést podrobnou analýzu a seznámit se s podobně zaměřenými programy. V případě, že by již existující projekt splňoval všechnu požadovanou funkčnost, nemělo by význam vytvářet novou aplikaci.

V této kapitole se budu také věnovat rozboru struktury vědeckého článku a zaměřím se také na běžné složení jednotlivých odstavců v článku. Tento rozbor by měl přinést důležité informace, které později využiji při návrhu a implementaci aplikace.

1.1 Struktura vědeckého článku

Vytvářená aplikace má sloužit k tomu, aby pomáhala vytvářet vědecké články. Proto musí dodržovat určité standardy, které jsou pro tyto články vyžadovány. První částí musí být abstrakt. Ten shrnuje hlavní aspekty článku a při jeho čtení se čtenář rozhodne, jestli má pro něj význam číst článek celý. Abstrakt by neměl být příliš dlouhý a neměl by obsahovat citovaný a odborný text. Abstrakt bývá volně dostupný, i když je přístup k celému článku zpoplatněn, a využívá se jako popis v internetových databázích vědeckých článků.[4]

Po abstraktu následuje úvod. Úvod rozvádí důvody, proč autor článek vytváří, a definuje problém, kterému se bude věnovat.[5] Může obsahovat i nejdůležitější informace z následujícího textu a odkazy na podobné práce. Autor v úvodu často uvádí konkrétní otázky, které by chtěl obsahem článku zodpovědět.[4]

Po uvedení článku by autor měl zpracovat oblasti, které se obecně nazývají Materials and Methods (Materiály a metody) a Results and Discussion (Výsledky a diskuze). Tyto oblasti obsahují informace o autorově výzkumu, získaných výsledcích, použitých metodách atd. Sekce s výsledky práce je vů-

bec nejdůležitější v celém článku. Autor v této části pečlivě analyzuje získané informace.[5] V této sekci jsou vítané tabulky nebo grafy, které čtenářům zpřehlední autorovu analýzu.[6]

V sekci diskuze by autor měl interpretovat výsledky s ohledem na to, co se o daném tématu vědělo dříve.[5] Strukturu a počet těchto částí určuje autor.

Celý článek je obsahově ukončen závěrem. Závěr by měl shrnovat nejzásadnější přínosy článku. Logicky uzavírá obsah článku. Autor říká, jestli zjištěné výsledky splnily očekávání a jak přispěla práce ke zkoumání problematiky v dané oblasti. Autor může do závěru zahrnut také to, jak bude ve výzkumu pokračovat.[6]

Uvedené části článku představují obsah, který chce autor svou prací sdělit. Aby byl článek kompletní, měl by obsahovat výčet klíčových slov - pojmů, které specifikují dané téma a reference - odkazy na citované zdroje.

1.1.1 Struktura odstavce

Výše popsaná struktura celého článku říká, jak má článek vypadat, ale podává málo informací pro efektivní našeptávání vět nebo bloků textu v závislosti na oblasti, která se aktuálně v odstavci edituje. Aby bylo našeptávání co nejúčinnější, je potřeba věty nabízet právě podle toho, do jaké části odstavce je autor potřebuje vložit.

Každá část článku se dá logicky rozdělit na jednotlivé typy vět. Toto rozložení je vždy stejné a liší se pouze počet vět daného typu.

Existují čtyři základní typy vět. Jako první jsou v odstavci umístěny Topic sentences(tematické věty).[7] Těch sice může být v odstavci víc, ale standardně se využívá pouze jedna. Tato věta slouží k uvedení odstavce a měla by v ní být vyjádřena hlavní myšlenka, proč je tento odstavec do článku vložen. Účelem tematické věty je říct čtenáři, co bude v následujícím textu probíráno.[8]

Po tematické větě následují Supporting sentences – věty, které mají podpořit, popřípadě upřesnit a rozšířit tematické věty. Do odstavce přinášejí detaily a vysvětlují důležité termíny.[7]

Concrete Detail sentences podobně jako Supporting sentences podporují a vysvětlují téma určené prvními větami. Měly by být zaměřené na jednu konkrétní oblast, která byla v odstavci zmíněna dříve. Concrete Detail sentences a supporting sentences tvoří hlavní obsahovou část článku.[7] Někdy jsou concrete detail sentences brány jako součást Supporting sentences.[9]

Closing sentences slouží k ukončení odstavce. Toto ukončení většinou probíhá jednou nebo dvěma větami. Uzavírací věty by neměly přinášet novou informaci, ale shrnout nebo jinak ukončit předešlý text.[7]

1.1.2 Kategorizace vět

V předchozí části jsem psal o rozdělení odstavců na různé typy vět podle toho, kde v odstavci se nacházejí a jakou mají v odstavci funkci. Nyní se zaměřím

na to, jak u vět identifikovat, do které části článku se daná věta hodí.

Jsou dvě části článku, pro které je možné pozorovat často opakující se věty nebo fráze. Tyto části jsou abstrakt a závěr. Pro anglický jazyk, ve kterém budou psané články, vytvářené pomocí vznikající aplikace, je velmi dobře popsané, které věty je možné využívat pro abstrakt a závěr.[7]

Například následující seznam obsahuje neosobní slovesné fráze, vhodné pro použití v závěru článku, a je získaný z knihy **The St. Martin's Handbooke**[7].

- It appears to/that
- It would appear to/that
- It seems to/that
- It would seem to/that
- It tends to be
- There is a tendency to/for
- It is said that
- Some of the evidence shows that
- Some writers say that
- It has been suggested that
- It is generally agreed that
- It is widely accepted that
- It is not generally recognised that

Díky tomuto a podobným seznamům je možné vytvořit rozsáhlejší databázi vět vhodných pro našeptávání tak, aby uživatel vybíral minimálně pro části článku abstrakt a závěr pouze z nabízených vět. Tyto seznamy jsou také vhodné pro vytvoření parametrizovaných vět, u kterých by uživatel pro umístění do článku musel upravit nebo doplnit určitou část tak, aby věta dávala v kontextu smysl.

1.2 Porovnání existujících řešení

Prvním nutným krokem analýzy je prozkoumání implementace stávajících řešení. Existují pro to dva důvody – inspirace pro další funkcionalitu a vyvarování se chyb či nedostatků, které jsou v těchto programech přítomny.

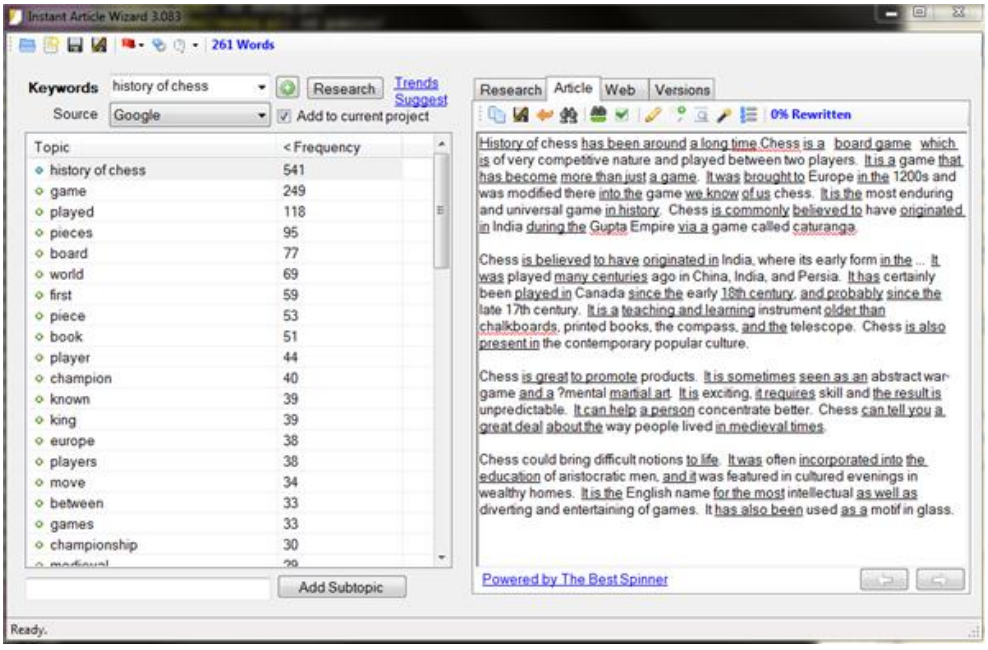
Existuje velké množství programů, které se zabývají tvorbou článků. Většina z nich obsahuje pouze jednodušší funkce, jako například automatické připravení referencí, ale neřeší samotnou tvorbu textu.

Dále existují řešení, která se snaží nabídnout uživateli kompletně vytvořený článek, jenž je sice unikátní, ale uživatel nemá moc možností, jak ovlivnit jeho obsah.

1.2.1 Instant article wizard

Nejzajímavějším řešením ze zde představovaných programů je komerční Instant Article Wizard. Jeho funkce se nejvíce blíží zadání této práce, i když pracuje na jiném principu. Program sám neobsahuje žádné předpřipravené fráze a vše se snaží najít online pomocí vyhledávačů jako je Google, Bing nebo Yahoo.[10] Celkem je možné vybírat z 10 různých vyhledávačů, pomocí kterých se informace vhodné k umístění do článku vyhledávají. Program se snaží najít fráze podle zvoleného tématu. Tyto fráze nabízí uživateli, který má možnost celou tuto frázi nebo její část vložit do článku. Na rozdíl od následujících dvou řešení, která podporují pro tvorbu článku pouze angličtinu, Instant Article Wizard podporuje jazyků šest – angličtinu, španělštinu, francouzštinu, němčinu, italštinu a portugalštinu.[10] Další rozdíl oproti následujícím řešením je, že výsledkem je pouze text místo kompletního článku. Na obrázku 1.1 je vidět, jak probíhá tvorba článku. V horní části uživatel vybere klíčové slovo a vyhledávač, v dolní části vlevo se ukazují výsledky vyhledávání a vpravo uživatel může zobrazit obsah jednoho nalezeného odkazu nebo text psaného článku.

Funkce vyhledávání informací na internetu pomocí vyhledávačů se v tomto programu jeví jako velice užitečná, a proto jsem se rozhodl podobnou funkci zabudovat i do programu, který bude výsledkem této práce.



Obrázek 1.1: Instant article wizard [1]

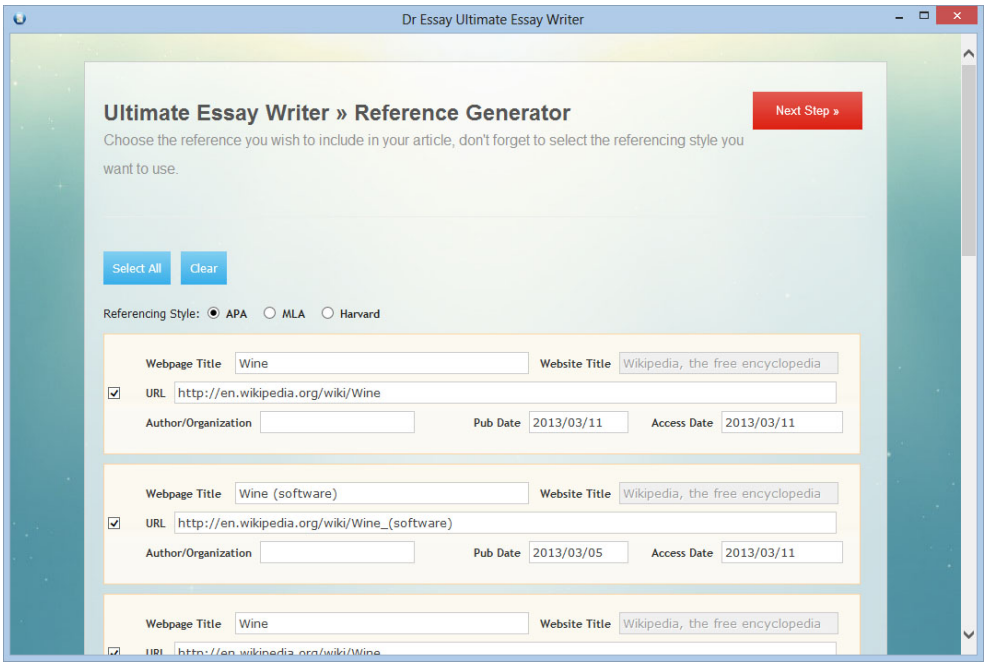
1.2.2 SCIgen

SCIgen je projekt, který vznikl v roce 2005 na MIT a je volně dostupný jako webová aplikace.[11] Umožňuje uživateli vygenerovat kompletní náhodný článek. Nevýhodou tohoto řešení je, že není možné nijak ovlivnit, o čem nový článek bude. Přesto byl SCIgen vyvíjený více než 10 let a některé vygenerované články byly odeslány na vědecké konference, jako například WMSCI (World Multiconference on Systemics, Cybernetics and Informatics).[11]

SCIgen jsem zařadil mezi podobná řešení k mé diplomové práci proto, že dokáže skládat souvislé a srozumitelné části textu, udržovat strukturu článku nebo například použitým větám přidávat citace. Tyto schopnosti by měl mít i program, který bude výsledkem této práce.

1.2.3 Dr Essay

Dr Essay je sada nástrojů, která umožňuje nejen generovat články podobně jako to umí SCIgen, ale snaží se i pomáhat při tvorbě článků vlastních. Zajímavou částí této služby je nástroj nazvaný Research Assistant. Ten vyhledává informace, které by bylo možné použít při vytváření článku podle vložených klíčových slov.[2] Mezi další funkce patří generátor citací, který je zobrazen na snímku 1.2, nebo nástroj nazvaný Article Rewriter. Ten se snaží nahradit často používaná slova ve vloženém textu slovy stejného významu.[2]



Obrázek 1.2: Dr Essay generátor citací [2]

1.2.4 Microsoft Word

Všechna předchozí řešení se zabývala čistě tvorbou článku, ať už ho vytvářela celý nebo pouze pomáhala při jeho tvoření nabízením textu. Jako poslední řešení jsem se rozhodl uvést program od společnosti Microsoft s názvem Word. I když MS Word a podobné kancelářské programy zastávají jiné funkce než aplikace, která je cílem této práce, přesto se dá říct, že se snaží autorovi zjednodušit práci při psaní textu. Jejich pomoc spočívá v jednoduchém formátování textu nebo například v detekci překlepů a špatného slovosledu.

1.3 Požadavky

Stanovení požadavků je proces stanovení služeb, které by měl systém poskytovat, a omezení, za nichž musí pracovat. Analýza požadavků je kritická ve vztahu k úspěšnému dokončení softwarového projektu. Požadavek musí být proveditelný, měřitelný a testovatelný. Také by měl být definovaný dostatečně detailně pro účely návrhu systému.[12] Požadavky jsou funkční a nefunkční.

1.3.1 Funkční požadavky

V této sekci popíši funkční požadavky. To jsou požadavky, které jsou kladeny na systém z pohledu uživatele, to znamená, co uživatel v systému může provádět.[12]

F1: **Systém bude umožňovat registraci nového uživatele** Uživatel bude moci vytvořit nový účet pomocí celého jména, zvoleného uživatelského jména a hesla.

F2: **Systém bude umožňovat přihlášení uživatele** Registrovaný uživatel se bude moci přihlásit pomocí uživatelského jména a hesla.

F3: **Systém bude umožňovat vybírat témata článku** Uživatel bude moci vybrat témata, která se budou týkat jeho článku.

F4: **Systém bude umožňovat vybírat klíčová slova** Uživatel bude moci zvolit klíčová slova pro upřesnění zvolených témat.

F5: **Systém bude umožňovat výběr nabízených vět** Uživatel bude moci vybrat a případně odmítnout nabízené věty.

F6: **Systém bude umoňovat vložit vlastní nové věty** Uživatel bude moci vložit vlastní věty do systému.

F7: **Systém bude umožňovat obnovení uloženého článku** Registrovaný uživatel bude moci obnovit dříve uložený článek pro pokračování v editaci.

F8: **Systém bude umožňovat vytvoření vědeckého článku** Uživatel bude mít možnost vytvořit vědecký článek.

F9: **Systém bude umožňovat formátování textu** Uživatel bude moci formátovat vytvořené části článku.

F10: **Systém bude umožňovat vkládání příloh** Uživatel bude mít možnost k napsanému textu vložit přílohu.

F11: **Systém bude umožňovat export článku** Uživatel bude mít možnost uložit článek, který vytvořil, ve formátu docx nebo tex.

F12: **Systém bude umožňovat editaci vytvořeného článku** Uživatel bude mít možnost upravit dříve vytvořený článek.

F13: **Systém bude umožňovat uložení článku** Registrovaný uživatel bude mít možnost vytvářený článek uložit.

1.3.2 Nefunkční požadavky

Nefunkční požadavky jsou takové požadavky, které jsou kladeny na systém z hlediska provedení.[12]

N1: **Systém bude dostupný přes web** Systém bude dostupný jako webová aplikace a nebude vyžadovat instalaci žádné speciální aplikace.

N2: **Články budou tvořeny v anglickém jazyce** Systém bude uživateli všechny věty a části textu nabízet v anglickém jazyce.

N3: **Systém bude všechna data ukládat do databáze** Systém bude věty, celé články, informace o uživatelích a další informace ukládat do relační databáze.

N4: **Systém bude poskytovat intuitivní ovládání** Systém bude uživateli pro ovládání nabízet přehledné a intuitivní rozhraní pro pohodlné ovládání aplikace.

N5: **Klientská a serverová část budou komunikovat pomocí REST api** Serverová část bude vystavovat RESTové rozhraní, pomocí kterého bude klientská část přistupovat ke stavům aplikace.

1.4 Případy užití

Use case diagram popisuje chování z pohledu uživatele a stanovuje, jaké má systém obsahovat funkce. Analýza případů užití se skládá z aktérů, případů užití (use case) a vztahů mezi nimi. Každý use case poskytuje jeden nebo

více scénářů, které zaznamenávají, jak by systém měl spolupracovat s koncovým uživatelem.[13] Kromě textového popisu jsou případy užití znázorněny na diagramu 1.3.

1.4.1 Aktéři

Aktér je role, která komunikuje s jednotlivými případy užití. V této roli může být obsazen uživatel, nebo externí systém.[13] V mé aplikaci budou dva základní typy uživatelů - registrovaný uživatel a host. Obě role se od sebe liší možností ukládání a obnovení článků. Další aktér bude správce systému. Jeho hlavní úlohou bude správa a plnění databáze.

1.4.2 Případy užití

Zaregistrovat se

Aktéři: Neregistrovaný uživatel

Scénář

1. Neregistrovaný uživatel zadá své registrační údaje.
2. Systém vytvoří neregistrovanému uživateli nový účet.

Přihlásit se

Aktéři: Registrovaný uživatel

Scénář

1. Registrovaný uživatel zadá své přihlašovací údaje.
2. Systém ověří správnost přihlašovacích údajů.
3. Systém umožní přihlášenému uživateli práci na článku.

Zvolit téma článku

Aktéři: Registrovaný uživatel, Neregistrovaný uživatel

Scénář

1. Systém nabídne uživateli seznam dostupných témat.
2. Uživatel vybere téma.

Zvolit klíčová slova

Aktéři: Registrovaný uživatel, Neregistrovaný uživatel

Scénář

1. Systém nabídne uživateli seznam dostupných témat.
2. Uživatel vybere téma.
3. Systém nabídne klíčová slova podle zvolených témat.

4. Uživatel vybere klíčová slova.

Uložit vytvořený článek

Aktéři: Registrovaný uživatel

Scénář

1. Registrovaný uživatel vytvoří článek.
2. Systém uloží článek.

Obnovit uložený článek

Aktéři: Registrovaný uživatel

Scénář

1. Registrovaný uživatel pošle požadavek na zobrazení všech svých článků.
2. Systém požadavek zpracuje a odešle seznam uložených článků.
3. Uživatel vybere jeden z článků a odešle požadavek na jeho obnovení.
4. Systém obnoví vybraný článek.

Výstup: Obnovený článek, který umožňuje další editaci.

Stáhnout článek jako TeX soubor

Aktéři: Registrovaný uživatel, Neregistrovaný uživatel

Scénář

1. Uživatel pošle požadavek na stažení článku.
2. Systém převede vytvořený článek do formátu TeX.
3. Uživatel stáhne vytvořený soubor.

Výstup: Dokument ve formátu TeX, který obsahuje vytvořený článek.

Parametrizovat větu

Aktéři: Správce systému

Scénář

1. Správce systému odešle požadavek na zobrazení nově přidaných vět.
2. Systém zpracuje požadavek a odešle správci systému seznam vět.
3. Správce parametrizuje vybrané věty.
4. Systém uloží nově vložené věty.

Vložit novou větu do databáze

Aktéři: Registrovaný uživatel, Neregistrovaný uživatel, Správce systému

Scénář

1. Uživatel nebo správce systému odešle do systému novou větu.
2. Systém uloží větu do databáze a přidá k ní informace získané z kontextu (typ věty, klíčová slova atd.).

Vybrat větu

Aktéři Registrovaný uživatel, Neregistrovaný uživatel

Scénář

1. Uživatel pošle požadavek na vygenerování vět.
2. Systém vybere věty podle zvolených témat a odešle je uživateli.
3. Uživatel vybere jednu nebo více vět.

Aktualizovat pořadí vět

Aktéři: Registrovaný uživatel, Neregistrovaný uživatel, Správce systému

Scénář

1. Uživatel odešle věty do systému v určitém pořadí.
2. Systém aktualizuje záznamy o každé větě.

Vytvořit článek

Aktéři: Registrovaný uživatel, Neregistrovaný uživatel

Scénář

1. Uživatel vybere věty a výběr odešle do systému.
2. Systém vybrané věty zpracuje a umožní uživateli provést zformátování textu.
3. Uživatel provede změny v textu nebo uloží článek.

Přiřadit větu ke klíčovému slovu

Aktéři: Správce systému

Scénář

1. Správce systému odešle požadavek na přiřazení věty ke konkrétnímu klíčovému slovu.
2. Systém zpracuje požadavek a uloží informaci do databáze.

Stáhnout článek jako Word dokument

Aktéři: Registrovaný uživatel, Neregistrovaný uživatel

Scénář

1. Uživatel pošle požadavek na stažení článku.
2. Systém převede vytvořený článek do formátu docx.
3. Uživatel stáhne vytvořený soubor.

Výstup: Dokument ve formátu docx, který obsahuje vytvořený článek.

Zobrazit uložené články

Aktéři: Registrovaný uživatel

Scénář

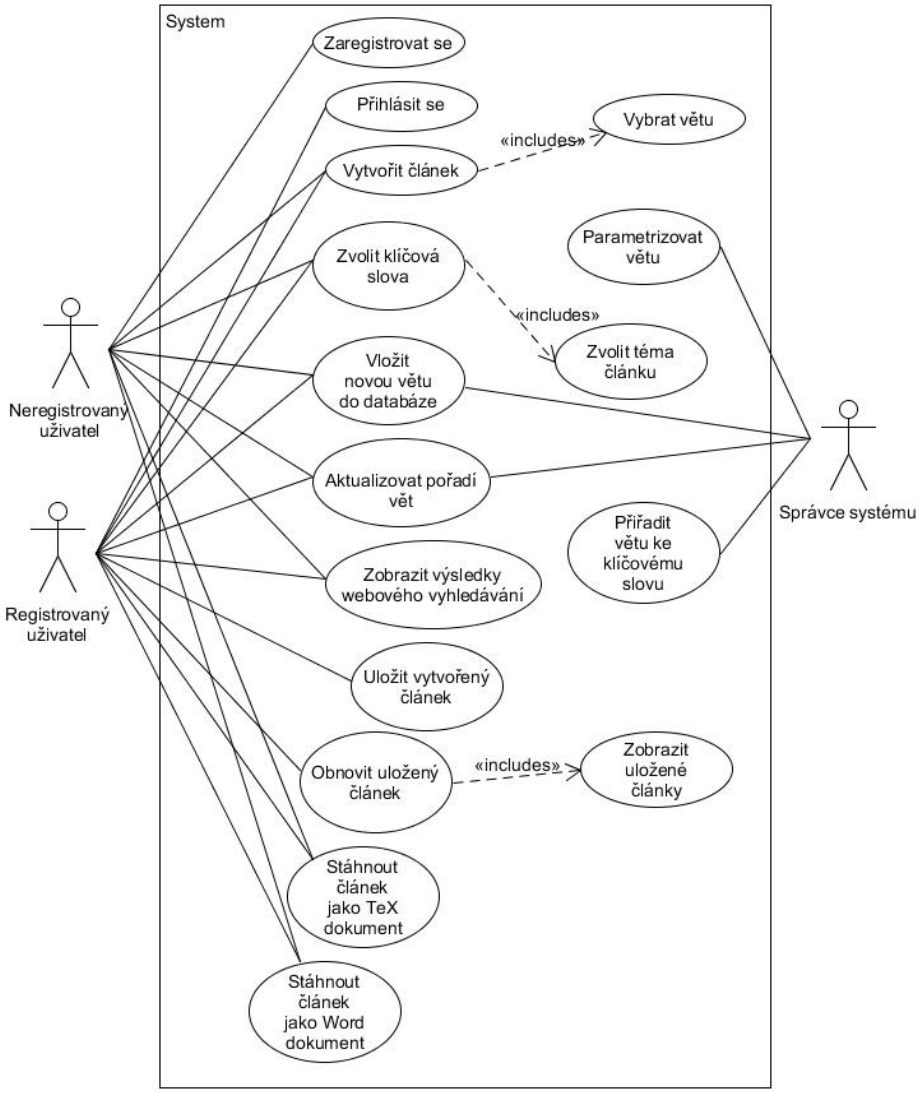
1. Uživatel pošle požadavek na zobrazení svých článků.
2. Systém požadavek zpracuje a odešle uživateli seznam článků.
3. Uživatel zobrazí své články.

Zobrazit výsledky webového vyhledávání

Aktéři: Registrovaný uživatel, Neregistrovaný uživatel

Scénář

1. Uživatel přidá nové věty do článku a odešle je do systému.
2. Systém se pokusí pro každou větu nalézt výsledky na internetu a odešle je uživateli.
3. Uživatel zobrazí výsledky vyhledávání.



Obrázek 1.3: Případy užití

Kapitola 2

Návrh

V předchozí kapitole jsem psal, jaké funkce by aplikace měla obsahovat. Aby bylo možné všechny tyto funkce implementovat, je nutné nejdříve celou aplikaci navrhnout a všechny funkce detailně popsat. K tomu slouží tato kapitola, ve které vysvětlím architekturu aplikace a popíši technologie, které budou při implementaci využity. Pokusím se navrhnout postup, jakým bude aplikace pomáhat autorům při tvorbě jejich článků. Detailně také popíši jednotlivé části aplikace.

2.1 Architektura

V této kapitole se pokusím popsat návrh jednotlivých komponent výsledného řešení. Celá architektura systému je znázorněna na diagramu 2.1. Jedná se o vícevrstvou architekturu, tvořenou třemi vrstvami - prezentační, aplikační a datovou.

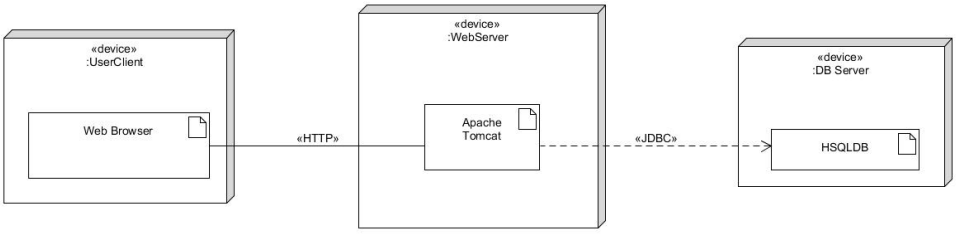
Hlavním bodem celého systému je server s nainstalovaným aplikačním serverem Apache Tomcat tvořící aplikační vrstvu. Na aplikačním serveru Apache Tomcat bude nainstalována vyvinutá aplikace a bude tvořit backend systému. S tímto serverem bude klient komunikovat z webového prohlížeče pomocí protokolu http a REST api.

V klientské části nebudou probíhat žádné výpočty, ale pouze zobrazování dat přijatých ze serverové aplikace. Bude tak sloužit jako prezentační vrstva.

Další částí systému bude databázový server s nainstalovanou relační databází HSQLDB. Aplikační server bude s databází komunikovat pomocí rozhraní JDBC. Databáze představuje datovou vrstvu systému.

2.2 Prezentační vrstva systému

Jak jsem zmínil výše, aplikace bude typu klient/server. V navržené třívrstvé architektuře bude klient představovat prezentační vrstvu (viz 2.1) a bude slou-



Obrázek 2.1: Diagram nasazení

žit čistě jako zobrazovací a ovládací prostředek, ve kterém nebude umístěna žádná logika aplikace. Tuto myšlenku ideálně splňuje webová aplikace, která přijímá informace ze serveru a zpět odesílá akce vykonané uživateli.

Tato část bude tvořena pomocí html a javascriptu. Pro jednodušší práci s kaskádovými styly, které ovlivňují výsledný vzhled webové stránky, jsem se rozhodl použít Bootstrap. Bootstrap je sada stylů a nástrojů pro tvorbu webu. Obsahuje velké množství možností, jak upravovat jednotlivé html elementy. Bootsrap patří mezi nejrozšířenější frameworky pro vývoj webových stránek.[14]

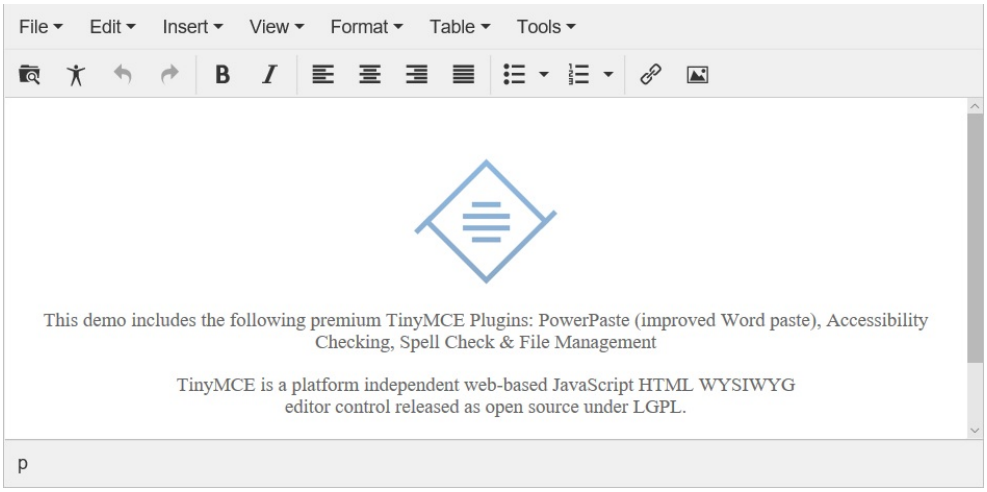
Aplikace také bude obsahovat takzvané WYSIWYG (What you see is what you get - co vidíš, to dostaneš)[15] okno pro dodatečnou editaci, vkládání příloh a formátování textu. K tomu bude použita javascript knihovna TinyMce, která dokáže udělat z běžného html elementu textarea wysiwyg editor. TinyMce je možné použít zdarma s omezeným počtem zásuvných modulů. Dá se využít i placená verze, kde jsou na výběr dvě varianty a každá nabízí oproti volné verzi více zásuvných modulů. Pro účely této práce zcela vyhovuje bezplatná verze.

2.3 Aplikační vrstva systému

K doplnění klientské webové aplikace bude existovat serverová část, která bude řešit veškerou logiku aplikace a bude vystavovat RESTové rozhraní pro komunikaci s klientskou částí aplikace.

Protože bude aplikace vyvíjena v jazyce JAVA, bude na fyzickém serveru nainstalovaný server aplikační, ve kterém poběží samotná aplikace. Jako aplikační server jsem zvolil APACHE TOMCAT, který je podrobně popsán v sekci 2.9.5.

Aby se mohla webová aplikace připojit k serveru a získat z něj potřebná data, musí mít aplikace vystavené RESTové rozhraní. To bude realizováno pomocí JAVA servletů.



Obrázek 2.2: TinyMce [3]

2.4 Datová vrstva systému

Pro efektivní a snadnou práci s daty je potřeba využít databázi, která pro vývoj a testování aplikace bude umístěna na stejném fyzickém stroji jako aplikační server, ale kdykoli může být jednoduše přesunuta. V práci bude využita volně dostupná relační databáze HSQLDB. Pro komunikaci mezi JAVA aplikací a databází bude využito rozhraní JDBC.

2.5 Návrh uživatelského rozhraní

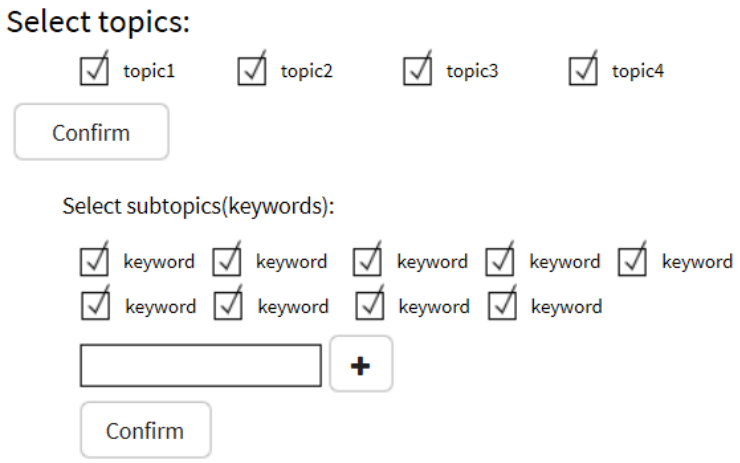
Přívětivé a přehledné uživatelské rozhraní je důležité pro snadnou práci s jakoukoli aplikací. Abych toho docílil, vytvořil jsem nejprve prototyp uživatelského prostředí – wireframe. Wireframe (drátěný model) se používá při návrhu webových stránek a aplikací a definuje rozmístění funkčních prvků. Nejedná se o finální grafický návrh, neobsahuje obrázky a nemusí obsahovat barvy.[16]

Nejjednodušší způsob, jak vytvářet drátěné modely, je v ruce pomocí papíru a tužky. Tyto modely mají nevýhodu, která spočívá ve složitém předělávání a zpracovávání změn.[16] Proto vznikají specializované programy, které umožňují dělat kvalitní modely velmi blízké reálnému řešení s možností „proklikávání“ aplikace.[17] Mezi nejpopulárnější programy patří například Balsamiq nebo Pencil.

Ve své práci jsem pro návrh wireframu použil volně dostupný nástroj MockFlow. Ten jsem zvolil s ohledem na to, že pro návrh webové stránky podporuje použít styly frameworku Bootstrap, který chci využít při implementaci webového klienta. Prototyp jsem se nesnažil vytvořit 1:1, jak by měla vypadat výsledná aplikace, ale snažil jsem se pomocí něj zjistit, jak rozmístit

ovládací prvky tak, aby ovládání bylo jednoduché.

Do horní části formuláře jsem umístil výběr témat (obrázek 2.3). Výběr bude probíhat pomocí zaškrtávacích políček, takzvaných checkboxů.



Obrázek 2.3: Wireframe model temata

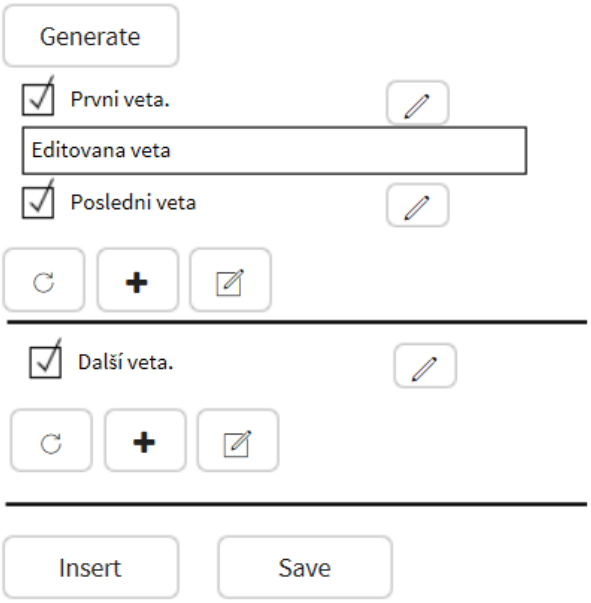
Pod výběrem hlavních témat je umístěný výběr podtémat – klíčových slov. Ty se zobrazí až po výběru jednoho nebo více témat uživatelem. Výběr klíčových slov bude opět probíhat formou zaškrtávání checkboxů.

Pod výběrem témat jsem umístil hlavní část aplikace. Ta má za úkol zobrazovat nabízené věty a zpracovávat uživatelem přidané věty. Věty se pro přehlednost budou zobrazovat pod sebou a graficky budou odděleny typy vět (obrázek 2.4).

Uživatel bude mít možnost nabídnutou větu upravit. K tomu bude sloužit tlačítko, které je na přiloženém výřezu modelu znázorněno symbolem tužky a je umístěno vedle každé věty. Každá oblast reprezentující jeden typ vět bude obsahovat tlačítko na přegenerování (aplikace odebere nevybrané věty a nahradí je novými) a další tlačítka na přidání jedné věty systémem nebo uživatelem.

Až bude uživatel s textem spokojený, použije tlačítko, které je na obrázku označeno textem Insert. Díky tomuto tlačítku přenese editování do editoru, který umožní formátovat text a vkládat přílohy. Tlačítkem Save uživatel uloží zformátovaný text, který se zobrazí v needitovatelné oblasti na konci webové stránky.

Pod oblastí sloužící pro formátování textu budou umístěna tlačítka pro export článku do formátů .docx a .tex.



Obrázek 2.4: Wireframe model věty

2.6 Návrh algoritmu našeptávání textu

Klíčovou částí celé aplikace je způsob výběru vět, ze kterých bude autor vytvářet svůj článek. I když uživatel musí hlavní myšlenku a teorii zformulovat sám, aplikace by mu měla inteligentně našeptávat příslušné fráze a věty, které by pak mělo být relativně jednoduché upravit nebo doplnit tak, aby přesně zapadaly do kontextu.

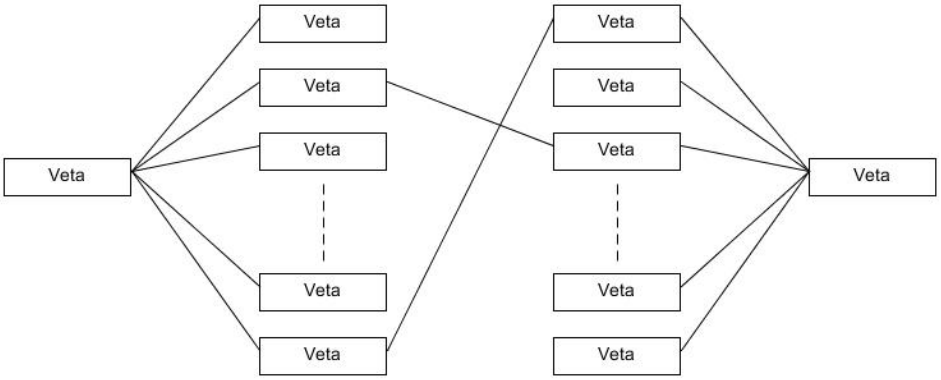
Základem pro našeptávání je kategorizace vět do témat. Autor by si měl při vytváření nového článku zvolit, čím se chce v článku zabývat. Tato volba probíhá ve dvou krocích. Jako první si autor zvolí obecná témata. Aplikace pak nabídne užší okruhy, po jejichž vybrání se opět upřesní množina vět, které mohou být nabídnuty pro vložení do článku. Našeptávání vět není závislé jen na zvoleném tématu, ale i na aktuálně psané části článku. Například pro abstrakt se využijí jiné věty než pro závěr. Dalším kritériem, které se musí zohlednit, je to, do které části odstavce se nová věta přidává.

Po zohlednění těchto kritérií jsem navrhl následující kroky, jak postupovat při našeptávání vět autorovi.

Při první iteraci systém seřadí věty podle priority a náhodně vybere zvolený počet vět tak, aby odpovídaly aktuálně tvořené části článku (universální věty) nebo vybranému tématu. Tyto věty jsou sice náhodné, ale jsou seřazené a rozdělené podle typu vět a vybírané podle priority, která se u každé věty vyvíjí podle toho, jak často ji nějaký autor využije pro svůj článek. U jednodušších a kratších částí se může v nejlepším případě stát, že autor již při

první iteraci vytvoří kostru aktuálně tvořeného odstavce. Pokud je tvořená část článku delší, uživatel může do vytvořené kostry vkládat vlastní text, popřípadě může dogenerovat další věty z databáze. Pokud si uživatel žádnou z vět nevybere, systém mu stejným způsoben nabídne věty jiné. Pokud si ale vybere alespoň jednu z nabízených vět, systém se bude snažit našeptávat věty, které by mohly vybranou větu logicky doplnit. V takovém případě se jako první kritérium bere, jestli existuje záznam o spojení vybrané věty s větami jinými. Na diagramu 2.5 je znázorněna struktura uchování spojení mezi větami. Tato spojení se aktualizují, přidávájí se nová a málo využívaná jsou uživateli nabízena s menší prioritou než spojení, která autoři častěji začleňují do svého textu.

Diagram 2.5 dále ukazuje, že pokud si autor článku vybere dvě věty, mezi které bude možné vložit další text, aplikace mu nabídne části textu, které by mohly vybrané věty logicky spojit. Díky tomuto výběru může uživatel pomocí několika kliků myší získat blok textu, do kterého nebude muset nijak zasáhnout.



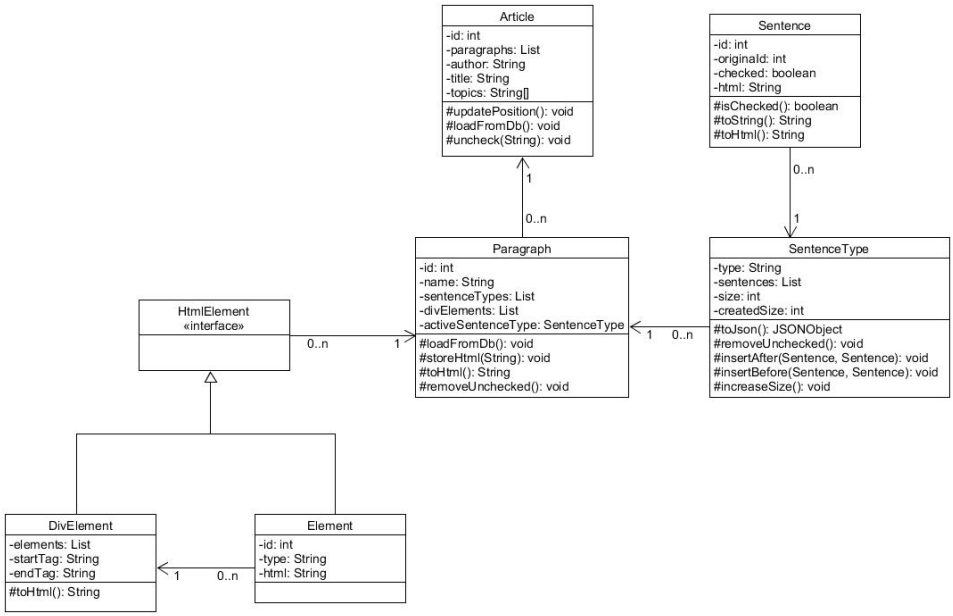
Obrázek 2.5: Našeptávání textových bloků

2.7 Model tříd

Na následujícím diagramu (2.6) je znázorněný model tříd odpovídající části programu, která slouží pro zpracovávání článku. Jak je z diagramu patrné, struktura je velice podobná databázovému modelu.

Při vytváření modelu tříd jsem úmyslně vynechal metody, které pouze vrací nebo nastavují hodnotu atributům. Nejvýše v modelu je třída Article, jejíž instance je vytvořena ve chvíli, kdy uživatel odešle příkaz pro obnovení článku z databáze, nebo vytvoření nového článku. Každý článek má id, které ho identifikuje pro databázi.

Článek obsahuje části, které reprezentuje třída Paragraph. Ta obsahuje metody pro převod vybrané části do html reprezentace a zpracování přijatého html, které obsahuje formátování textu, přílohy a nově přidaný text. Každá část článku se skládá z typů vět. O jejich zpracování se stará třída SentenceType. Každý typ obsahuje množinu vět (třída Sentence), stará se o přidávání nových vět, udržování jejich pořadí a odstraňování vět, které se uživatel rozhodl v článku nepoužít. Pro zpracování html využívá třída Paragraph interface HTMLElement a jeho implementace DivElement a Element. DivElement reprezentuje html tagy div, které obalují textové bloky a obsahují informace o formátování, například zarovnání textu na střed nebo velikost písma. Div elementy obsahují množinu ostatních tagů. Ty přestavuje třída Element. Pokud Element představuje větu, obsahuje její id, pomocí kterého může větu dohledat v konkrétním typu vět. Objekt typu Element kromě id obsahuje atributy type a html. Podle atributu type se rozeznává, který html tag Element zastupuje. Atribut html obsahuje kompletní reprezentaci elementu ve formátu html.



Obrázek 2.6: Class diagram

2.8 Návrh datové vrstvy aplikace

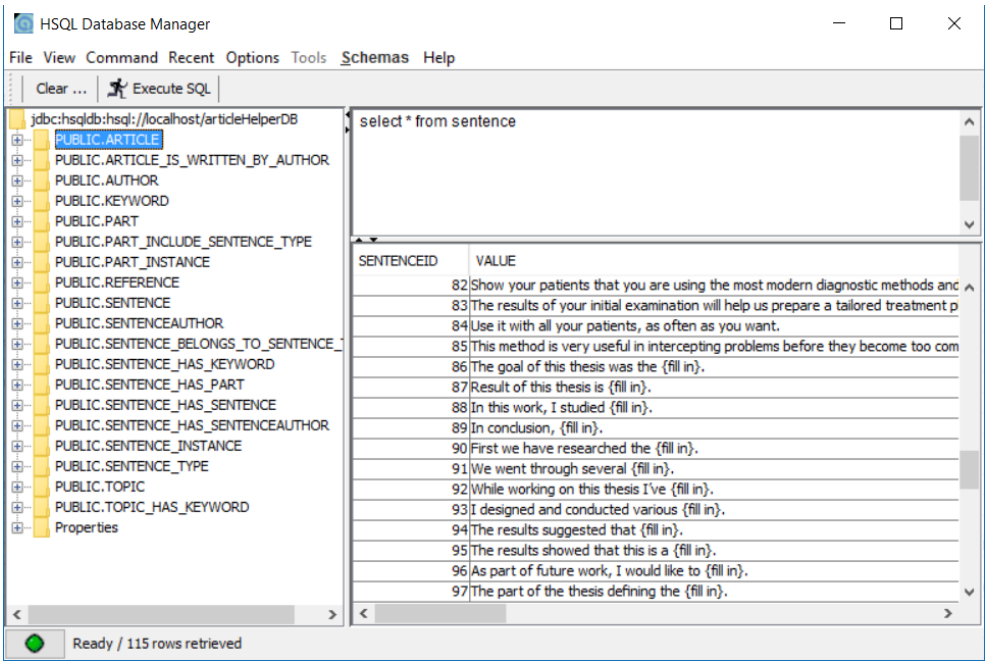
Stejně jako algoritmus našeptávání vět je pro funkčnost aplikace důležitá zásoba vět, ze kterých je možné vybírat, jejich uložení a provázání mezi sebou.

Aby bylo možné data uložit, vyhledávat a třídit, je výhodné využít vlastnosti relační databáze. Je tak možné použít její funkčnost a není nutné psát vlastní vyhledávací algoritmy.

2.8.1 Výběr databáze

Pro uložení dat jsem pro aplikaci vybral relační databázi Hyper SQL Database, která je obecně označována zkratkou HSQLDB. Tato databáze je založena na platformě Java. To jí umožňuje multiplatformní využití. Mezi její největší výhody patří velmi malá velikost (HSQLDB zabírá na disku přibližně 20 MB) a jednoduchá instalace.[18]

HSQLDB nabízí pro správu databáze vlastní manager, který dovoluje provádět SQL příkazy a zobrazovat strukturu tabulek.[19]

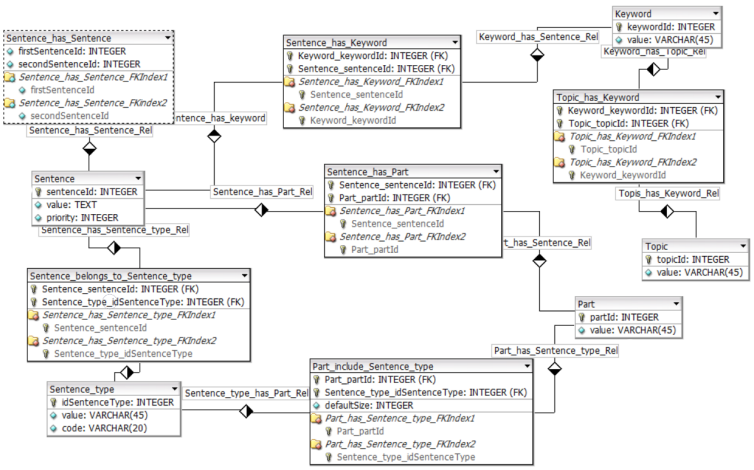


Obrázek 2.7: HSQLDB manager

2.8.2 Návrh datového modelu

Navržený datový model je možné rozdělit na tři části. První část tvoří tabulky pro uložení samotných vět a dalších informací a dat potřebných pro funkci vytváření obsahu článků. Druhá část reprezentuje uživatele - autory a jejich články, které vytvořili. Poslední třetí část tvoří podpůrné informace k jednotlivým větám, které umožňují vkládat do článku citace k větám nebo blokům vět.

První, nejdůležitější část databáze, je tvořena tabulkami, které reprezentují informace potřebné pro vytváření datového obsahu článků. Tabulka Topic



Obrázek 2.8: Model ukládání vět

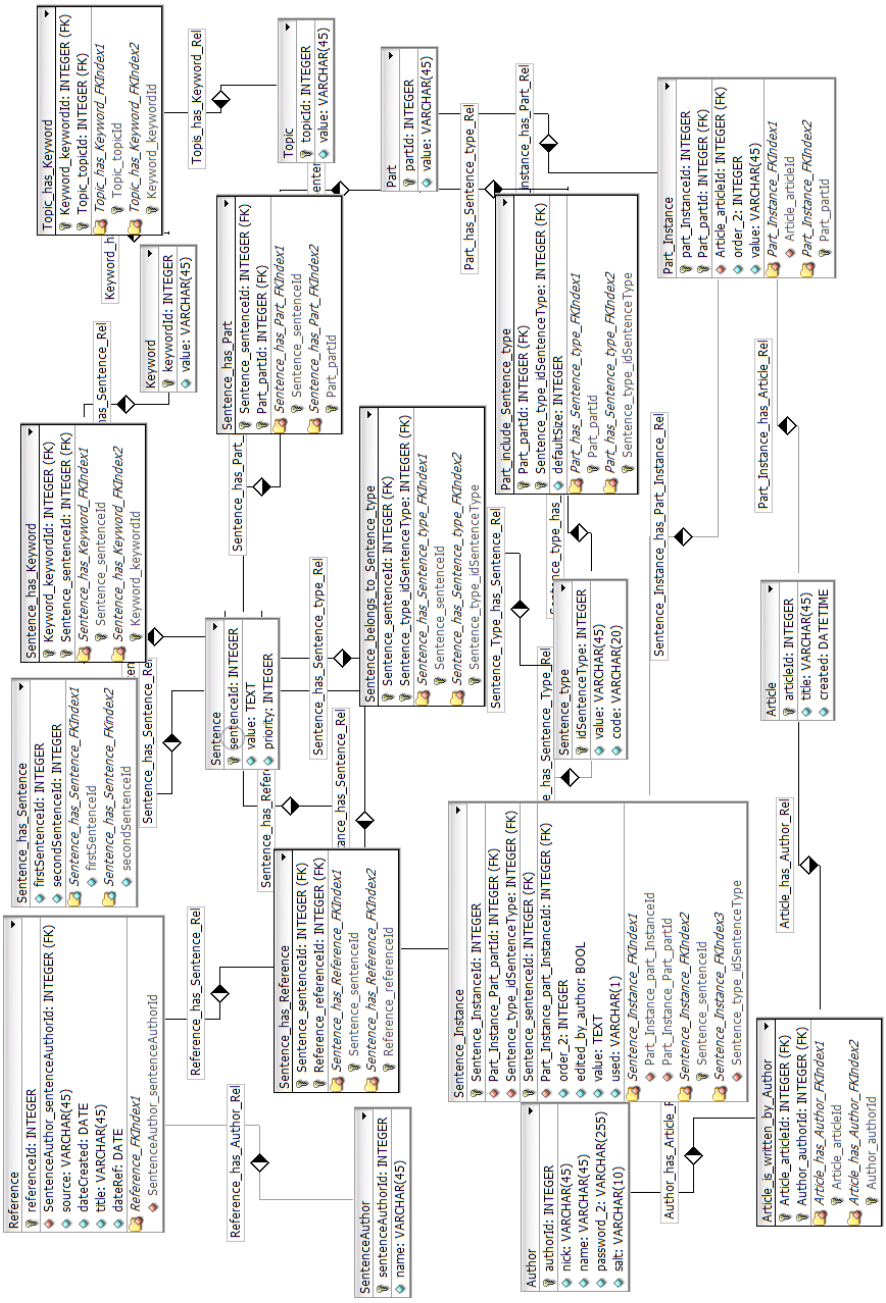
reprezentuje obecná témata, ze kterých si může autor článku vybrat a určit tím, o čem by článek přibližně měl být. Ke každému tématu se vztahují klíčová slova (tabulka Keyword), která zpřesňují, co bude obsahem článku. Pod jednotlivými klíčovými slovy už je možné hledat konkrétní věty. Věta (Sentence) může patřit pod žádné nebo pod několik klíčových slov. Nabízení vět je dále zpřesněno pomocí určení, jakého je věta typu. Díky tomu je možné alespoň přibližně určit, ve které části odstavce je možné větu použít. Typy vět jsou určeny částí článku (například úvod). Každá část má přednastavené složení, které uživatel může změnit.

Druhá oblast představuje autory a vytvářené nebo hotové články patřící autorům. Pro autora si systém pamatuje jeho přihlašovací údaje a jméno, které se uvádí v článku. Pro článek se ukládá jeho název a datum vytvoření. Na článek se navazují jednotlivé instance částí, které autor vytvořil. V těch se uchovávají veškeré informace o vzniklém článku, například přidané věty nebo formátování odstavce v html. Pro uložení konkrétní věty, kterou autor přidal do článku, sloučí tabulka Sentence_instance. Pro každou větu, kterou autor do článku přidá, se vytvoří její instance. Tato instance existuje, i pokud autor větu z textu odebere. To se využívá pro zpřesnění našeptávání vět a úpravu priorit, se kterými jsou věty uživateli nabízeny. Pokud autor článku vybírá větu na místo, pro které již existují instance vět, aplikace se snaží najít věty jiné.

Poslední část doplňuje k větám informaci o jejich zdroji a slouží pro vytváření citací. Pokud je pro větu citace vyžadována, ukládá se její autor, zdroj a v případě webových stránek datum vytvoření citace.

Jak vypadá schéma celé databáze po poskládání těchto částí, je znázorněno

na diagramu 2.9.



Obrázek 2.9: Model databáze

2.9 Použité technologie

Tato kapitola popisuje technologie, které byly využity při návrhu a implementaci aplikace. Jsou zde obsaženy všechny důležité prvky, kterých aplikace využívá, jako jsou protokoly, rozhraní nebo datové formáty.

2.9.1 JDBC

JDBC je zkrácený název pro Java Database Connectivity a reprezentuje jednotné rozhraní pro přístup k relačním databázím. JDBC vzniklo jako vrstva mezi databází a JAVA aplikací. Tato vrstva dokáže získávat data z databáze jako instance JAVA tříd a vkládat instance JAVA tříd do SQL. Pro přístup ke konkrétnímu databázovému serveru musí být k dispozici specifický JDBC ovladač, který zpravidla poskytuje tvůrce databáze.[20]

2.9.2 http

Hypertext Transfer Protocol – zkráceně http – je protokol, který byl původně vytvořen pro přenos dokumentů ve formátu HTML, ale v současné době se využívá i pro přenos dalších informací.[21] Protokol definuje několik metod, které se využívají pro komunikaci. Metody se nazývají GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS a CONNECT. Nejpoužívanějšími jsou první 4 jmenované.[21]

Zpráva se skládá z hlavičky a dat. Hlavička může obsahovat informace o odesílateli zprávy, datum vytvoření zprávy, typ přenášených dat, nebo například stavový kód volání.[21] Stavové kódy říkají, jestli bylo volání úspěšné nebo neúspěšné. Pokud proběhla komunikace úspěšně, stavový kód je obvykle **200 OK**. V opačném případě bude kód z množiny kódů Client Error (4XX) nebo Server Error (5XX).[22] Data mohou být v libovolném formátu, například html, xml nebo JSON. Protokol http neobsahuje žádné zabezpečení, proto existuje zabezpečená verze https. Ten pro zabezpečení spojení využívá protokol SSL (Secure Sockets Layer).[21] V poslední době je doporučováno využívat u všech webových stránek právě zabezpečený https protokol místo nezabezpečeného a snadno napadnutelného http.

2.9.3 REST

Representational state transfer (zkráceně REST) je koncept pro design distribuované architektury.[22] Na rozdíl například od protokolu SOAP (Simple Object Access Protocol - protokol pro výměnu zpráv založených na XML), je REST orientován datově a ne procedurálně. Všechny zdroje mají vlastní identifikátor URL, který určuje také stav aplikace. Tento princip se nazývá HATEOAS-Hypermedia as the Engine of Application State.[22]

K přenosu dat využívá transportního protokolu http a jeho operací GET, POST, PUT a DELETE. Protože ale Java servlety, které využiju při imple-

mentaci RESTového rozhraní, podporují pouze operace GET a POST, omezím se ve vytvářené aplikaci na tyto dvě operace.

2.9.3.1 GET

GET je dotazovací operace, kterou využívají například webové prohlížeče při získávání obsahu webových stránek. Dotaz se definuje pomocí URL, vrácená data jsou předávána v rámci těla http odpovědi. Metoda GET patří mezi takzvané bezpečné operace, protože při jejím zavolání nedochází na serveru k žádným změnám. Podobně se chovají také metody HEAD a OPTIONS.[21]

2.9.3.2 POST

Na rozdíl od metody GET využívá POST k vytváření dotazu primárně tělo zprávy. Tam umístí data, která mají být zpracována na serveru. Zpracování dat může na serveru způsobit změny, například odstranění nebo přidání nějakého záznamu, proto operace POST není bezpečná.[21] I když by POST měla sloužit primárně pro úpravu existujících záznamů na serveru a pro vytváření a mazání by měly být využívány operace PUT a DELETE, často je, hlavně u webových aplikací, kvůli zjednodušení pro všechny tyto úlohy využívána pouze operace POST.[21]

2.9.3.3 Struktura URL

URL, celým názvem Uniform Resource Locator, je řetězec znaků s definovanou strukturou, který slouží ke specifikaci umístění zdrojů informací. Typicky se využívá jako identifikátor webových stránek.[23]

URL se skládá z názvu protokolu, který je pro komunikaci využíván, identifikátoru serveru, portu, umístění zdroje na serveru a formulářových dat.[23]

Pro komunikaci je v této práci využíván protokol http, proto bude každá URL odkazující na zdroje serveru začínat *http://*. Port může být libovolný. Pro webové stránky se nejčastěji využívá port 80. V takovém případě se port nemusí do URL zadávat. Pro aplikační server Apache Tomcat je standardní hodnota portu 8080.[23]

V aplikaci je každý zdroj na serveru vytvářen pomocí samostatného JAVA servletu. Identifikátor zdroje odpovídá názvu třídy, která daný servlet implementuje.

Po identifikátoru zdroje následuje formulář dat. Ten je od zbytku URL oddělen znakem **?** a je obecně nazýván jako query řetězec. Právě tato část URL je klíčová pro vytváření požadavku na server. Query řetězec se skládá z hodnot klíč=hodnota a ty jsou odděleny znakem **&**.[23]

2.9.4 JSON

Json(JavaScript Object Notation) je datový formát určený pro přenos dat, která mohou být organizována v polích nebo agregována v objektech. Je čitelný pro člověka. Alternativou pro JSON je například XML, která se ale moc nehodí pro webové aplikace, vzhledem k tomu, že obsahem zprávy je přibližně 40% značek a jejich atributů.[24]

Objekty jsou ve formátu JSON tvořeny polem typů klíč: hodnota. Klíč musí být řetězec v uvozovkách. Hodnota může být číslo, řetězec, boolean, null a pole. JSON se stal standardem pro použití v RESTu.

```
{
  "prvni": 1,
  "druha": 1.1,
  "treti": "abc",
  "ctvrta": null,
  "pata":
  [
    1,
    2
  ]
}
```

V klientské části aplikace je JSON zpracováván pomocí jazyka JavaScript, který umí JSON zpracovávat nativně, a práce s ním je velmi jednoduchá a intuitivní.

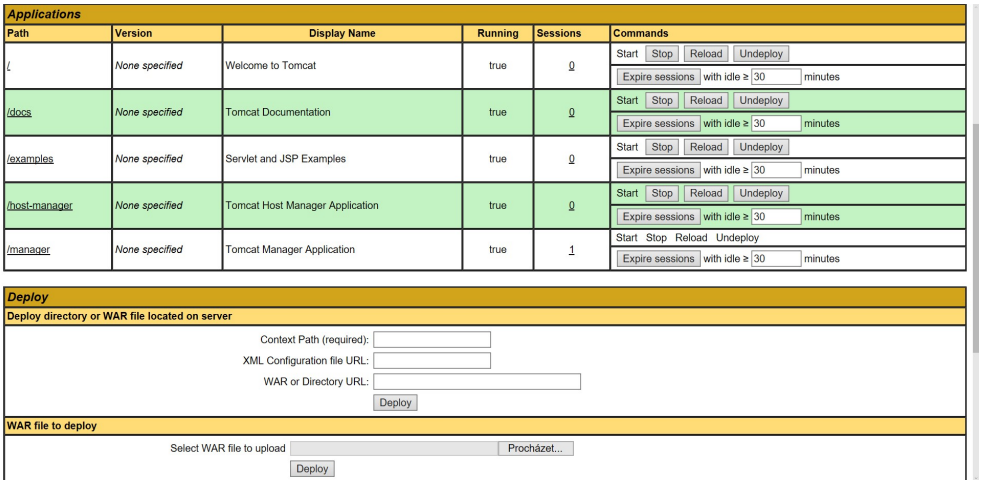
Na serverové části je JSON zpracováván pomocí jazyka Java, který pro práci s tímto formátem potřebuje externí knihovnu. Ve své práci jsem využil knihovnu org.json. JSON objekt je reprezentován třídou JSONObject a pole třídou JSONArray.

2.9.5 APACHE TOMCAT

Protože je práce koncipovaná jako klient/server a serverová část je tvořena pomocí jazyka Java, je potřeba pro provoz aplikace aplikační server. Aplikační server je software specializovaný pro provoz sdílené aplikace. Výhodou použití aplikačního serveru je, že vývojář může mnohem více času věnovat vývoji logiky aplikace, a nemusí se tolik zabývat infrastrukturou aplikace, jako například zabezpečením.[25]

Pro svoji práci jsem vybral aplikační server od společnosti Apache Software Foundation, který má název Tomcat verze 8.5. Apache Tomcat je open-source aplikační server a Java Servlet Container. Je založený na Jave a podporuje většinu Java EE specifikací jako Java Servlet, Java Server Pages, Java EL a WebSocket.[26]

Tomcat je velmi oblíbený pro svoji jednoduchost, stabilitu, otevřenost a podporu komunity.



Obrázek 2.10: Webová konzole pro správu aplikačního severu Apache Tomcat

2.9.6 JAVA HTTP Servlet

HTTP Servletem se v jazyce JAVA nazývá každá třída, která implementuje interface javax.servlet.http.HttpServlet. Tato třída zpracovává http požadavky a pro tyto účely musí obsahovat implementace metody doGet, která zpracovává http operaci GET, a doPost, která zpracovává http operaci POST. Servlety poskytují platformě nezávislou metodu pro vytváření webových stránek.[27]

Největšími výhodami servletů je bezpečnost, která vychází z bezpečnostních funkcí podporovaných jazykem Java, rychlost zpracování http požadavků a jednoduchost vývoje.[28]

Java servlety jsou využívány v další technologii, která umožňuje vytvářet dynamické webové stránky. Tato technologie se nazývá JSP (JavaServer Pages). Ta umožňuje omezit množství javascript kódu v aplikaci tím, že se do html kódu píše logika v jazce Java. Z tohoto html dokumentu se následně vygeneruje výsledná stránka.[27]

2.10 Výhody řešení

Aplikace je navržená tak, že ji uživatel obsluhuje pomocí webového klienta a veškerá logika aplikace včetně databáze je umístěna na serveru. Díky tomu mohou být aplikace i databáze jednoduše centrálně aktualizovány, aniž by si uživatel musel nějaké aktualizace stahovat a instalovat. Kromě toho tím odpadá tvorba samostatných aplikací pro různé operační systémy. Uživatel může k aplikaci přistupovat z jakéhokoli zařízení, které obsahuje webový prohlížeč.

2.11 Omezení řešení

Zvolená architektura má kromě výše psaných výhod také nevýhody. Především tu, že uživatel musí mít neustále k dispozici internetové připojení, pomocí kterého klientská část komunikuje s částí serverovou. Ne každému uživateli také musí vyhovovat zpracování řešení jako webové aplikace a upřednostil by samostatný desktopový program.

Z hlediska samotného vytváření článků je velkým omezením hlavně malý počáteční počet vět, který by se ale měl s častým používáním aplikace rozšiřovat. Poměrně složitým úkolem je kategorizace vět a textových bloků, které je nutné začlenit do správných kategorií a témat a udržovat povolené vztahy mezi nimi.

Kapitola 3

Realizace

V této kapitole podrobně rozeberu vybrané části implementace aplikace. Zaměřím se především na klíčové funkce a některé zajímavé postupy, které jsem při implementaci využil.

Popíši způsob komunikace mezi klientskou a serverovou částí aplikace, způsob přihlášení uživatele a uchování spojení mezi webovým prohlížečem a serverem, nebo způsob exportu výsledného článku do formátů .docx a .tex. Většina popisovaných metod bude ukázána na konkrétních příkladech.

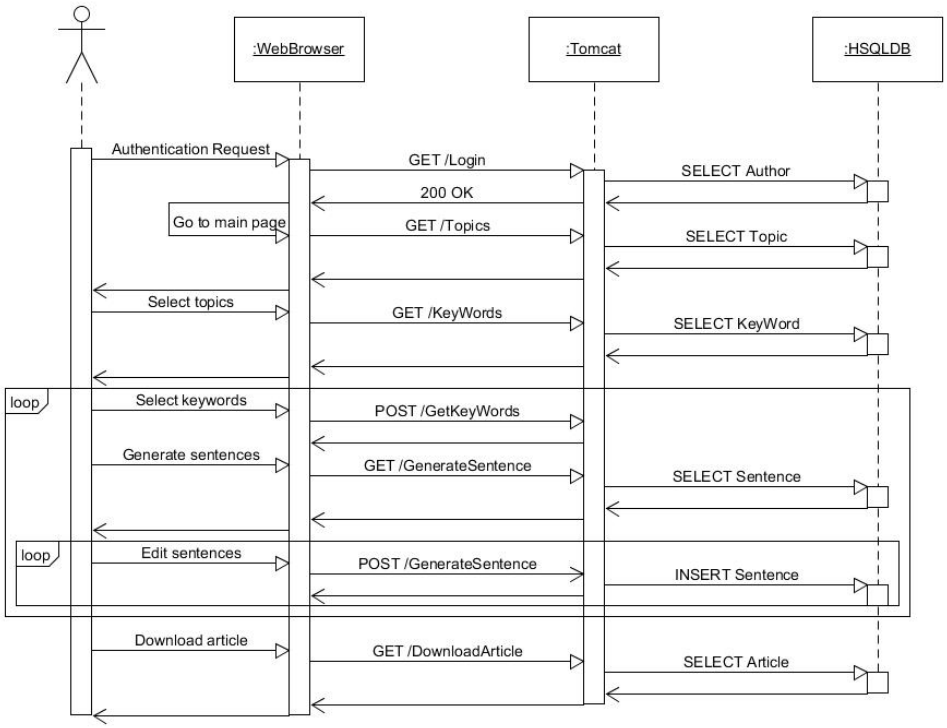
3.1 Sekvenční diagram

Na diagramu 3.1 je zjednodušeně znázorněna komunikace mezi jednotlivými částmi systému společně s příkazy uživatele. Uživatel příkazy zadává pomocí webové aplikace, která běží ve webovém prohlížeči. Ten příkazy přenáší pomocí HTTP protokolu a jeho operací GET a POST na stranu aplikačního serveru. Aplikační server obsahuje logiku aplikace a využívá JDBC rozhraní pro komunikaci s databází. Databáze obsahuje definovaná data a ukládá práci uživatelů.

Zobrazený sekvenční diagram se dá logicky rozčlenit na tři části. První část je v podstatě přípravná. Zaregistrovaný uživatel zadá přihlašovací údaje, ty se pomocí http GET a http basic authentication odešlou na server, tam jsou ověřeny a zpět je poslána informace o úspěchu nebo neúspěchu přihlášení. Pokud jsou přihlašovací údaje správné, klientská část aplikace obdrží http kód 200 a je přesměrována na stránku, kde jsou zobrazeny uložené články, a uživatel má možnost jeden z nich obnovit a pokračovat v editaci, nebo vytvořit nový. Část s výběrem článku není na sekvenčním diagramu znázorněna.

Sekvenční diagram pokračuje navigací webového klienta na hlavní stránku, na které probíhá tvorba článku. Při načtení této stránky je odeslán dotaz pomocí operace GET na zobrazení témat, pro které jsou v databázi dostupné věty vhodné pro vytvoření článku.

V prostřední části probíhá vytváření článku. Diagram znázorňuje opakované upřesňování klíčových slov. Ta jsou původně nabídnuta podle zvolených témat a uživatel je může měnit podle toho, čím se v konkrétní části článku zabývá. Klient posílá dotaz na nabídnutí nových vět operací GET na adresu /GenerateSentence. Na stejné adrese se zpracují nové věty, které uživatel přidá. K tomu slouží operace POST. V závěrečné fázi pošle uživatel dotaz na stažení vytvořeného článku.



Obrázek 3.1: Sekvenční diagram

3.2 Rest api

V této sekci podrobně popíši podobu RESTového rozhraní a způsob komunikace a vyměňování informací mezi klientskou webovou aplikací a aplikačním serverem.

V sekci 2.9 jsem zmínil, že u technologie REST reprezentuje URL stav aplikace. To znamená, že pokud to umožňuje délka a struktura dotazu, je přenášen pomocí query části URL. Některé specifické informace jako například sessionId jsou přenášeny v hlavičce http volání a ostatní data jsou přenášena

ve formátu JSON v těle zprávy. Pokud odpověd serveru obsahuje data, jsou opět přenášena v těle zprávy.

Typickým příkladem, kdy klient pro přenos dat využije pouze query část URL, je při volbě témat a klíčových slov. Pro nastavení témat posílá webový klient http POST požadavek na adresu */Topics*. Webový klient poskládá query výraz z vybraných témat vytvořením dvojic topic=“tema“ a spojením znakem &.

Pokud uživatel vybere témata *Mathematics* a *Algorithms*, celý dotaz vypadá takto:

```
/Topics?topic=Mathematics&Algorithms
```

Data ze serveru jsou zpět webovému klientovi posílána ve formátu JSON v těle zprávy. Následující ukázka zobrazuje zprávu, která je klientovi odeslána při přegenerování jednoho typu věty.

```
{
  "part": "Abstract",
  "type1": [
  {
    "id": 11,
    "checked": true,
    "sentence": "The goal is to show that the {fill in}."
  },{
    "id": 12,
    "checked": false,
    "sentence": "This work demonstrates {fill in}."
  }]
}
```

Zpráva obsahuje informace o části článku, kterou uživatel vytváří, do kterého typu věty patří a ke každé větě identifikátor a příznak, jestli ji uživatel již vybral.

Klientská aplikace jen nepřijímá data ze serveru, ale odesílá zpět informace o uživatelově činnosti. K tomu dochází pomocí http operace POST. Data jsou přenášena nejčastěji v případě, kdy uživatel upraví text věty, změní pořadí vět, nebo provede úpravu textu nebo formátování ve wysiwyg editoru.

Následující ukázka znázorňuje odeslání zformátovaného textu z webové aplikace na server. Html kód je odesílaný kompletní, aby nedošlo ke ztrátě dat o formátování, a je následně zpracován na serverové části aplikace. Z ukázky je také patrné, jak probíhá formátování textu a identifikace vložených vět. Blokový element ** ** je zarovnán na střed přidáním stylu *style="text-align: center;"* a věty jsou součástí tagu ****, který má jako id identifikátor konkrétní věty v aplikaci.

```
{
"part":"Introduction",
```

```
"text":"
<!DOCTYPE html>
<html>
<head></head>
<body>
  <div style="text-align: center;">
    <span id=\"st_d1505\">Bubble sort is a
  simple and well-known sorting
  algorithm.
    </span>
    <span id=\"st_d1518\">There are several
  ways to implement the bubble sort.
    </span>
  </div>
</body>
</html>"
}
```

3.3 Přihlášení

Autor má k aplikaci uživatelský účet, který mu umožňuje ukládat a znovuotvírat dříve vytvořené články. Pro přihlášení je v aplikace využita metoda **http basic authentication**. Tento způsob autentikace uživatele je u webových služeb velmi často využívaný pro jednoduchou implementaci a skutečnost, že jej podporují všechny internetové prohlížeče.[32] Jeho nevýhodou je, že jméno a heslo je odesíláno v nezašifrované podobě, a proto může být relativně snadno zcizeno, pokud se pro komunikaci nevyužívá zabezpečený https protokol.

Uživatelské jméno a heslo je během přihlašování odesláno na stranu serveru v nezašifrované, pouze zakódované podobě. Ke kódování se využívá metoda Base64. Pomocí té se zakóduje spojení uživatelské_jméno:heslo. Tento kód je přenesen pomocí http hlavičky **Authorization: Basic**. Celá zpráva požadující přihlášení uživatele se jménem *username* a heslem *password* vypadá takto:

```
GET /Login HTTP/1.0
Host: localhost:8080
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

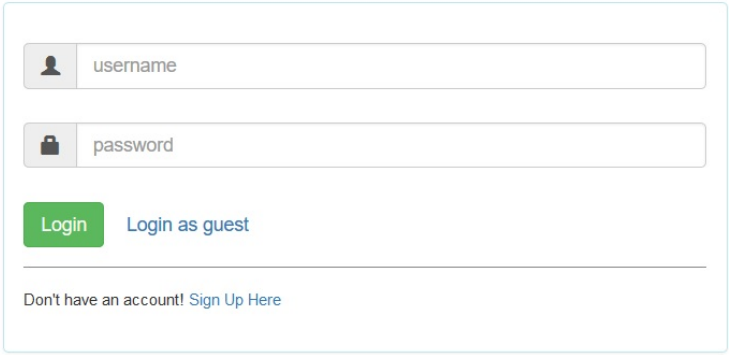
Pokud je uživatel úspěšně ověřen, je zpět odeslána zpráva s http kódem 200 OK, který informuje uživatele o úspěchu autentizace. V opačném případě je stavový kód 401 Unauthorized.

Odpověď serveru na úspěšné přihlášení má následující podobu:

```
HTTP/1.1 200
cookie: 541966B8E797C512615ABE530A2597F4
```

Date: Mon, 10 Apr 2017 18:57:59 GMT

Hlavička cookie obsahuje unikátni identifikátor - sessionId. Ten je spojení přidělen po přihlášení a slouží pro identifikaci operací, které uživatel v rámci systému provádí. Protože je protokol http bezstavový (neumí uchovávat informace o dřívější komunikaci), musí být sessionId uložen v rámci webového prohlížeče. Toto uložení trvá 1 den, nebo dokud se uživatel z aplikace neodhlásí.



Obrázek 3.2: Přihlašovací obrazovka

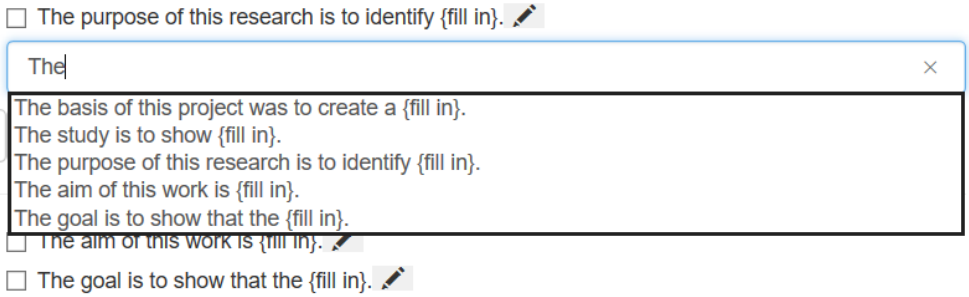
3.4 Automatické doplňování vět

Uživatel má možnost, kromě vybírání z nabízených vět, také přidávat do článku věty vlastní. Aby bylo možné i tuto činnost zjednodušit, je do aplikace přidána funkce automatického doplňování vět. Uživatel začne psát text a aplikace se mu snaží nabídnout pět nejpodobnějších výrazů, které má uložené v databázi.

Pro tuto funky bylo potřeba přidat do klientské aplikace funkci asynchronního volání http metody GET, aby aplikace neblokovala psaní textu uživatelem, ale přesto vyhledávala shody textu průběžně.

V klientské aplikaci je seznam takto nabízených vět řešen pomocí tagu **<datalist>**. Každá věta musí být do listu umístěna jako hodnota atributu *value* tagu **<option>**. Tento seznam se obnovuje vždy, když uživatel změní hodnotu v textové oblasti, kam právě doplňuje vlastní větu. Nová hodnota je pomocí asynchronní operace GET jako součást URL adresy přenesena na server a tam jsou vyhledány fráze, které obsahují shodnou část textu. Z těch je podle priority vybráno pět a ty jsou odeslány zpět do klientské aplikace.

Na snímku obrazovky 3.3 je vidět, jak toto nabízení vět vypadá.

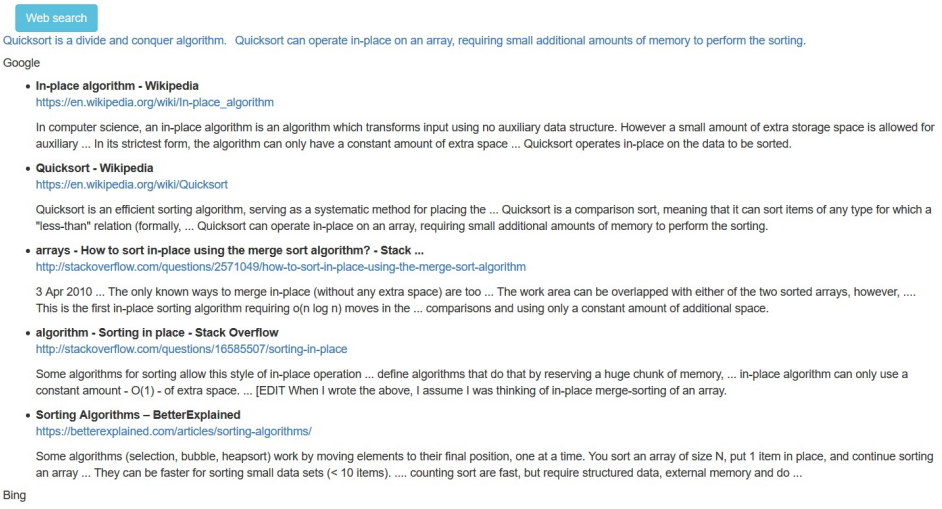


Obrázek 3.3: Automatické doplňování vět

3.5 Vyhledávání na webu

V sekci 1.2.1 jsem mluvil o programu s názvem Instant article wizard, který se snaží našeptávat části textu nalezené různými internetovými vyhledávači. Tato funkce se jeví jako velmi užitečná, a proto jsem se rozhodl její jednodušší variantu začlenit do tvořené aplikace.

Vyhledávání probíhá pomocí vyhledávačů Google a Bing a spustí se pro každou větu, kterou uživatel přidá nebo významně upraví. Aplikace se nejprve snaží najít přesné znění vložené věty. Tyto výsledky mohou uživateli posloužit jako nápověda, jak a čím pokračovat při psaní článku. Pokud vyhledávače nenaleznou žádné výsledky, aplikace začne spouštět vyhledávání výsledků tématicky odpovídajících dané větě.



Obrázek 3.4: Výsledek vyhledávání informací pomocí Google vyhledávače

Na snímku obrazovky 3.4 je znázorněná část html formuláře, která slouží pro zobrazení výsledků vyhledávání. Snímek vznikl při psaní krátkého zku-

šebního textu o řadícím algoritmu Quicksort, do kterého byly přidány věty "*Quicksort is a divide and conquer algorithm.*" a "*Quicksort can operate in-place on an array, requiring small additional amounts of memory to perform the sorting.*". Mezi jednotlivými výsledky může uživatel přecházet překlikáváním na odkazy reprezentovanými samotnými větami. Pro každou větu se zobrazí omezený počet výsledků s odkazem na cílovou stránku a s krátkým popisem.

I když oba vyhledávače podporují rozhraní pro strojové vyhledávání, tato rozhraní jsou velice omezená a je pomocí nich možné za jednu hodinu provést jen malý počet vyhledávání. Proto jsem se rozhodl využít standardní rozhraní vyhledávačů a výsledky získat z vráceného html kódu. Tato metoda je časově náročnější, ale je možné takto získat neomezený počet výsledků.

Oba vyhledávače podporují nastavení vyhledávání pomocí query řetězce. Celá url pak vypadá pro Google takto:

```
https://www.google.com/search?q=vyhledavany+vyraz&num=5
```

a pro Bing:

```
https://www.bing.com/search?q=vyhledavany+vyraz&count=5
```

Tato funkce byla do aplikace začleněna, aby ukázala možnosti, jak pomocí internetových vyhledávačů nabídnout autorům informace, které by mohli využít při tvorbě článku. Její funkčnost je velice jednoduchá, ale měla by sloužit jako základ pro přesnější a detailnější nabízení informací uživateli. Zlepšení této funkce by se dalo docílit například nezobrazováním jen nalezených odkazů, ale prohledáním obsahu nalezené stránky a nabídnutí konkrétních spojení, které by se hodily do tvořeného textu.

3.6 Citace

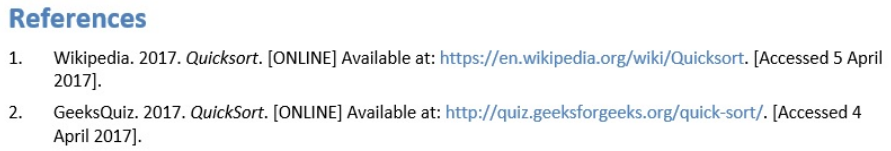
Citace zdrojů, ze kterých autor čerpal, jsou nedílnou součástí každého vědeckého článku, a proto s nimi musí umět pracovat i aplikace vytvářená v rámci této práce. Pro zpřehlednění webového klienta jsem se rozhodl, že citace budou součástí až hotového staženého článku a nebudou se zobrazovat při tvorbě článku.

S článkem aplikace pracuje ve formátu HTML, proto jsou i citace transformovány do tohoto formátu. Následující ukázka zobrazuje vytvořenou citaci pro internetovou encyklopedii Wikipedia. Pro vytvoření citace je využit tag **<blockquote>**, který je přímo určen pro tvorbu referencí od HTML5.

```
<blockquote>Wikipedia. 2017. <em>Quicksort</em>
  . [ONLINE] Available at:
  <u>
    <a href="https://en.wikipedia.org/wiki/Quicksort">
      https://en.wikipedia.org/wiki/Quicksort
```

```
    </a>
  </u>. [Accessed 5 April 2017].
</blockquote>
```

Snímek obrazovky 3.5 zobrazuje, jak vypadají vytvořené citace v článku uloženém do formátu Word.



Obrázek 3.5: Ukázka vygenerovaných citací

3.7 Export článku

Vytvoření článku by nemělo význam, kdyby si ho uživatel nemohl stáhnout a uložit ideálně v dále editovatelné a zpracovatelné podobě. Pro svoji práci jsem se rozhodl začlenit možnost exportu článku ve formátu docx – výchozím formátu pro Microsoft Word, a formátu tex, který je výchozím formátem pro program pro počítačovou sazbu.

3.7.1 Realizace exportu

Při exportu jsem se rozhodl využít toho, že uživatel článek tvoří ve formátu html, ve kterém jsou všechny informace o formátování, vložených obrázcích atd. S html umí výborně pracovat konzolový program pandoc, který dokáže převádět mezi sebou velké množství formátů, mimo jiné právě html, docx a tex. Pandoc je navíc opensource a je možné ho využívat zcela zdarma.

Díky jeho universálnosti proto není nutné psát vlastní převodníky do docx a tex, ale celou práci vykonají dva příkazy v příkazové řádce.

Pro Word:

```
pandoc -s article.html -o article.docx
```

Pro TeX:

```
pandoc -s article.html -o article.tex
```

3.8 Příprava textu

Vyvinutá aplikace nebude nikdy schopna efektivně napomáhat při tvorbě článků, pokud se nepodaří vybudovat dostatečně rozsáhlou databázi vět. Proces plnění databáze je časově velmi náročný. Je nutné projít velké množství

textu, vybrat jeho vhodné části, kategorizovat jednotlivé věty a vytvořit potřebné vazby mezi nimi.

Veškerý text použitý pro rozšíření databáze byl vybírán z veřejně dostupných zdrojů a pokaždé byl uložen i zdroj dat, který je využíván pro tvorbu citací. V době psaní tohoto textu byly připravené věty pro matematická témata integrál a derivace, řadící algoritmus Bubblesort a věty o přístrojích využívaných ve stomatologii Diagnocam a Cad/Cam. Pro možnost vytvoření abstraktu a závěru článku byly přidány věty, které nejsou tematicky zaměřené, ale jsou vhodnější pro umístění do těchto částí článku.

U některé věty je možné a také výhodné parametrizovat. Díky tomu mohou být použity ve více typech vět, částech článku, nebo pro více témat. Parametrizace byla potřeba hlavně u textu určeného pro abstrakt a závěr článku.

Aby bylo možné věty do databáze vložit, je potřeba vytvořit SQL skripty zajišťující vytvoření potřebných záznamů. Pro každé vkládané téma se vytváří pět skriptů. Z každého skriptu, vytvořeného pro vložení vět vhodných pro téma Bubble Sort, jsem vybral krátkou ukázku.

První slouží pro vložení samotných vět do tabulky **Sentence**.

```
INSERT INTO Sentence (value, priority) VALUES
('Average and worst case complexity of
bubble sort is O(n^2).', 5);
```

Druhý skript nově vložené věty přidá ke vhodnému tématu.

```
INSERT INTO Sentence_has_Keyword (Keyword_keywordId,
Sentence_sentenceId) VALUES
((select keywordid from keyword where value=
'Bubble Sort'),
(SELECT sentenceId FROM Sentence WHERE value=
'Average and worst case complexity of
bubble sort is O(n^2).'));
```

Třetí skript přiřadí věty do správných typů.

```
INSERT INTO Sentence_belongs_to_Sentence_type
(Sentence_sentenceId, Sentence_type_idSentenceType)
VALUES ((SELECT sentenceId FROM Sentence WHERE
value='Average and worst case complexity of
bubble sort is O(n^2).'), (SELECT idSentenceType
FROM Sentence_type WHERE code='type1'));
```

Čtvrtý skript přiřadí věty k vhodným odstavcům.

```
INSERT INTO Sentence_has_Part (Sentence_sentenceId,
Part_partId) VALUES ((SELECT sentenceId FROM
Sentence WHERE value=
'Average and worst case complexity of
bubble sort is O(n^2).'), (SELECT partId FROM Part
```

```
  WHERE value='Chapter'));
```

Poslední **pátý** skript zapíše do databáze vztahy mezi větami.

```
INSERT INTO Sentence_has_Sentence (firstSentenceId,
 secondSentenceId) VALUES ((SELECT sentenceId FROM
 Sentence WHERE value=
 'Average and worst case complexity of
 bubble sort is O(n^2).'),(SELECT sentenceId FROM
 Sentence WHERE value=
 'Also, it makes O(n^2) swaps in the worst case.'));
```

Kapitola 4

Testování

Během vývoje i po něm byla funkčnost a uživatelská přívětivost aplikace zkoušena sérií testů. Testy jsem prováděl od jednotkových testů, přes testování RESTového api až k testování uživateli. Každý z testů přinesl pro vývoj důležité poznatky a napomohl ke zlepšení aplikace.

V této kapitole nejprve popíši testování stability aplikace, rozeberu testování komunikace mezi serverovou a klientskou částí aplikace, otestuji uživatelské rozhraní pomocí heuristické analýzy a zhodnotím výsledky testování uživateli.

Na závěr jsem se rozhodl vložit jeden z článků vytvořených během uživatelského testování s podrobným popisem, jak vznikal a které jeho části musel autor napsat sám a které mu pomohla vytvořit aplikace.

4.1 Testování aplikace

Tato sekce popisuje prostředky a postupy, pomocí kterých jsem vyvinutou aplikaci testoval z hlediska stability kódu a správně pracujících požadovaných funkcí. Tato část testování je nezbytná, aby bylo možné zahájit testování pomocí testovacích uživatelů. Kdyby nebyla provedena, dostali by uživatelé neodladěnou aplikaci, u které by hrozily časté pády a velká nestabilita.

4.1.1 Jednotkové testy

Jednotkový test obvykle testuje pouze danou konkrétní jednotku. V ideálním případě by měl být každý testovaný případ nezávislý na ostatních, měl by testovat jednu konkrétní metodu, mělo by jich být mnoho a měly by být automatizované.[31] Výhodou unit testů je fakt, že když test selže, je velmi snadné lokalizovat chybu, protože každý test ověřuje jen malou část funkčnosti.

4.1.2 JUnit

Serverová část aplikace je napsána v jazyce JAVA, pro který je vhodný testovací framework JUnit. Všechny metody, které slouží ke spouštění testů, jsou označeny anotací @Test. V těchto metodách se porovnávají předpokládané výsledky s reálnými výsledky, které vrátí testované metody. Porovnávání výsledků probíhá pomocí metod, jejichž názvy začínají klíčovým slovem assert. Před spuštěním testů je možné provést instrukce označené anotací @Before.[29]

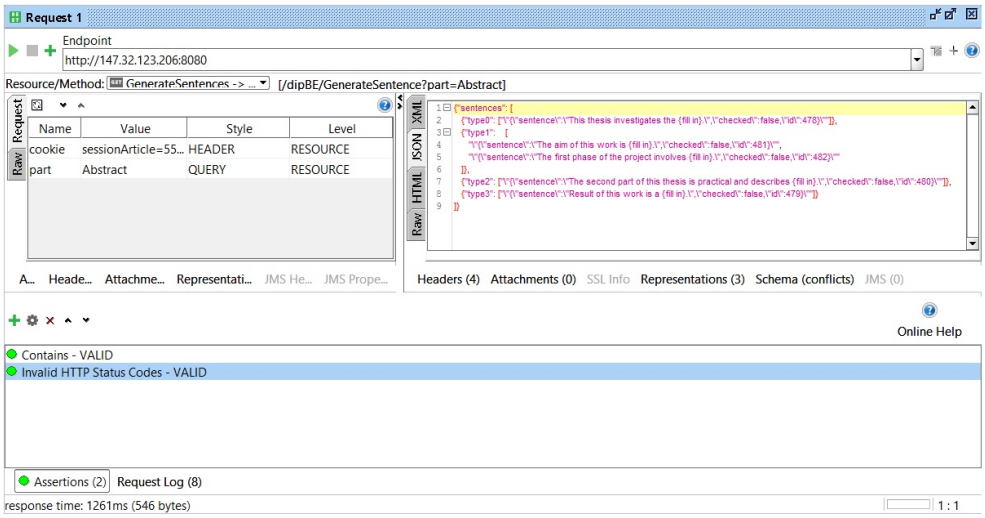
Aby bylo testování touto metodou účinné, měla by být testy pokryta většina veřejných metod. Tyto testy jsou spouštěny při každém sestavování aplikace, aby se předešlo přidání nových chyb do aplikace.

4.1.3 Testování REST api

Kromě otestování serverové části aplikace je zapotřebí otestovat komunikaci mezi serverovou a klientskou částí aplikace. K tomu se dá využít například nástroj SoapUI, ve kterém jsem připravil sérii testů. Tyto testy jsou založeny na provolávání serverové části a porovnávání očekávaných výsledků s reálně vrácenými hodnotami. Na rozdíl od JUnit testu je ale nutné, aby volání probíhala v určitém pořadí, a měla přesně stejnou podobu jako volání prováděná klientem aplikace. Takže při testech se například musí poslat požadavek na přihlášení uživatele předtím, než se odešle požadavek na zobrazení dostupných témat. Těmito testy jsem se snažil pokrýt kompletní funkčnost. Vytvořené testy kontrolují vrácený stavový kód, který se musí shodovat s očekávaným výsledkem. Správnost vrácených stavů jsem ověřoval pro úspěšné i neúspěšné zavolání požadavku-například chybně zadané přihlašovací údaje. Kromě stavových kódů je při testech kontrolován i obsah zpráv.

4.1.4 Klientská aplikace

Klientskou část aplikace, kterou tvoří HTML a javascript, jsem otestoval nejen na funkčnost a uživatelskou přehlednost a pochopitelnost, ale zaměřil jsem se i na otestování kompatibility s posledními verzemi hlavních webových prohlížečů-Google Chrome, Mozilla Firefox, Internet Explorer/Edge a Opera. V každém webovém prohlížeči jsem otestoval kompletní funkčnost aplikace a zjistil následující výsledek: Jako problémový prvek se ukázalo vkládání obrázků pomocí wysiwyg editoru TinyMce, které jsou uloženy na místním disku. V tomto případě jsou obrázky v prohlížeči převedeny do formátu base64 a tato reprezentace je zapsána jako zdroj(src) pro <img> tag reprezentující obrázek. Tuto funkci nepodporují prohlížeče Internet Explorer a Opera. Jako doporučené se proto dají označit prohlížeče Google Chrome a Mozilla Firefox, které podporují všechny funkce aplikace.



Obrázek 4.1: Testování REST rozhraní pomocí SoapUI

4.2 Testování uživatelského rozhraní

Komponentu uživatelského rozhraní má uživatel na očích po celou dobu používání aplikace. Uživatelské rozhraní rozhoduje často o tom, zda uživatel bude danou aplikaci používat či nikoliv. Musí být proto navrženo co možná nejpřehlednějším způsobem, aby se v něm uživatel necítil ztracen a používalo se mu intuitivně.

4.2.1 Heuristická analýza

Heuristická analýza je často využívaná a poměrně účinná metoda, která si dává za cíl přezkoumání uživatelského rozhraní pro zvýšení intuitivnosti. Díky této metodě se dá snadno zjistit porušení základních pravidel a zvyklostí definovaných pro tvorbu uživatelského rozhraní. Tím vzniká prostor pro opravení těchto nedostatků ještě před testováním reálnými uživateli.

Tuto metodu sestavil a popsal Jakob Nielsen, proto je často nazývána jako Nielsenova analýza.[30]

4.2.1.1 Viditelnost stavu systému

Uživatel je vždy informován o stavu systému a o skutečnosti, jakou operaci zrovna systém provádí.

Při provádění heuristické analýzy bylo ovšem zjištěno, že aplikace nedostatečně informuje uživatele při delším načítání dat ze serveru. Proto byla do aplikace přidána animace indikující čekání na odpověď serveru.

Obrázek 4.2: Ukázka animace načítání

4.2.1.2 Shoda mezi systémem a realitou

Uživatel bude od aplikace, která má za cíl zjednodušit psaní článku, očekávat formu textového editoru, která je mu nabídnuta pomocí wysiwyg textového elementu. K tomu má navíc možnost usnadnit si práci přidáváním nabídnutých textových bloků do vlastního textu.

V systému jsou také použity ikony, které korespondují s funkcionalitou z běžné praxe, například "plus" znamená přidat položku, nebo "tužka" znamená editovat.

4.2.1.3 Minimální zodpovědnost

Uživatel může vždy jednoduše odstranit změny, které v aplikaci udělal, jejich vymazáním v oblasti aplikace sloužící k formátování a úpravám textu. Stejně tak může změnit vybrané elementy (například témata nebo nabídnuté věty) kdykoli během vytváření článku.

Pokud bude chtít uživatel vrátit změny provedené v textovém editoru, může využít standartní klávesové zkratky **Ctrl+Z** a **Ctrl+Y**, nebo může využít tlačítek s šipkami zpět a dopředu.

Obrázek 4.3: Tlačítka pro vracení změn v textovém editoru

Během analýzy jsem zjistil, že aplikace nezachová změny v textu a formátování, pokud před uložením uživatel například obnoví stránku v prohlížeči.

4.2.1.4 Shoda s použitou platformou a obecnými standardy

Aplikace je implementována v prostředí webu a dodržuje vžité webové standardy.

4.2.1.5 Prevence chyb

V aplikaci nejsou zabudované funkce, které by dovolovaly uživateli v jednom kroku například smazat celý článek, proto nebylo nutné do aplikace přidávat

ověřovací dialogy, které by uživatel musel před provedením nebezpečné operace potvrdit.

4.2.1.6 Kouknu a vidím

Část aplikace, sloužící k tvorbě článků, je implementována jako jedna samostatná webová stránka. Ta je přehledně rozdělena na oblasti sloužící k výběru témat, editaci textu a zobrazení kompletního článku. Uživatel tak vždy ví, ve které části aplikace se pohybuje.

4.2.1.7 Flexibilita a efektivita

Uživatel si může vybrat, jak bude aplikaci využívat. Může používat pouze textový editor a napsat kompletní text sám, nebo může využít všechny pokročilejší funkce.

4.2.1.8 Minimalita

Aplikace se snaží být minimalistická a rozděluje uživatelské prostředí na části pro tvorbu článku, kde uživatel vidí vždy jen vybranou část informací, a zobrazení kompletního textu.

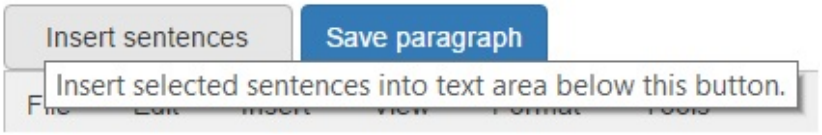
4.2.1.9 Smysluplné chybové hlášky

Aplikace se snaží zobrazovat kontextová chybová hlášení přímo u daných elementů, aby uživatel nebyl zmaten a ihned viděl, jak svou chybu opravit.

Obrázek 4.4: Chybová hláška při chybném zadání počtu vět

4.2.1.10 Help a dokumentace

Aplikace se snaží uživateli radit vyskakovacími dialogy při najetí myší nad ovládací prvek. Součástí aplikace je i podrobná uživatelská příručka, ve které je ovládání detailně popsané.



Obrázek 4.5: Ukázka popisu ovládacího prvku

4.2.1.11 Shrnutí

Během heuristické analýzy jsem objevil některé nedostatky uživatelkého prostředí, které jsem se pokusil vyřešit ještě před dalším průběhem testování. Nejviditelnější změnou je přidání indikace načítání při odeslání požadavku na server.

4.3 Uživatelské testování

Všechny předchozí testy, kromě heuristické analýzy, testovaly části aplikace a snažily se objevit chyby v naprogramování nebo chybně navržené komunikační rozhraní. Uživatelské testování má za cíl objevit chyby v rozvržení ovládacích prvků a ověřit použitelnost aplikace při tvorbě vědeckého článku.

Pro tento druh testování jsem sestavil testovací scénář. Ten obsahuje úkoly, které by měl každý uživatel provést. Úkoly jsem se snažil zvolit tak, aby uživatel musel využít co největší počet vložených funkcí. Díky tomu se dá ověřit, zda je uživatelům jasné, jak mají aplikaci použít, a jestli aplikace dokáže efektivně pomoci při tvorbě článku. Úkoly začínají vytvořením uživatelského účtu, provádějí uživatele výběrem témat a prací s větami a textovými bloky a nakonec uživatel stáhne vytvořený článek.

Formulář s testovacím scénářem dále obsahuje otázky, pomocí kterých mi uživatelé dají zpětnou vazbu o použití aplikace. Otázky se zabývají použitelností a rychlostí aplikace. Uživatelé mají také možnost do dotazníku napsat nápady na budoucí rozšíření funkčnosti aplikace.

4.3.1 Výsledky uživatelského testování

O testování jsem požádal deset uživatelů. Cílem bylo, aby testovací uživatelé nebyli zaměření na jeden konkrétní obor. Každý si vybral téma odpovídající svému oboru a pokusil se vytvořit krátký text, který by odpovídal požadavkům určených úkoly v dotazníku.

Po splnění úkolů zodpověděli testovací uživatelé otázky přiložené k dotazníku. Shrnutí těchto odpovědí je zobrazené v tabulce 4.1. Z výsledků vyplývá, že uživatelé byli spokojeni s ovládáním aplikace. Při maximální známce 10 byla průměrná hodnota odpovědi **9**. Kladně hodnotili i odezvu programu, věcnost nabízených vět a smysluplnost nabízených textových bloků.

Velmi důležitým zjištěním bylo, že testovací uživatelé byli schopni vytvořit díky aplikaci kompletní články a zjednodušení hodnotili mezi známkami **8** až **10**.

Otázka	Nejčastější	Průměr	Nejlepší	Nejhorší
Užitečnost nápovědy	8	8	8	7
Intuitivní ovládání	8	9	10	8
Věty odpovídají tématu	8	9	10	7
Smysluplné textové bloky	9	8	9	5
Odezva programu	9	9	10	6
Zjednodušení tvorby článku	8	9	10	8

Tabulka 4.1: Výsledky testování uživateli

V části dotazníku, kde měli uživatelé shrnout klady aplikace, uváděli rychlost tvorby článku, použitelnost nabízených vět, nebo například intuitivní ovládání, ke kterému nepotřebovali zobrazit nápovědu. V nedostatcích aplikace pak uváděli například omezené množství nabízeného textu nebo nemožnost zvolit si výchozí jazyk pro tvorbu článku. Testovací uživatelé také pomohli nalézt některé chyby v aplikaci. Jedna z těchto chyb byla neúplné vložení vytvořeného článku do stahovaného word dokumentu. Další chyba souvisela s ne příliš dobrou schopností wysiwyg editoru TinyMce pracovat s obrázky, které mají větší rozlišení, a obrázkové přílohy byly roztažené mimo okraje stránky.

Kromě toho testovací uživatelé také do dotazníku uvedli několik zajímavých typů na budoucí rozvoj aplikace. Jeden z nápadů navrhoval přesunout našeptávání vět pouze do textového editoru, kde by se během psaní zobrazovaly nabízené věty v oblasti pod kurzorem. Další typ navrhoval přidat do našeptávače možnost zobrazit seznam vět, které mohou být přidány po aktuálně vybrané větě bez nutnosti přegenerovat nevybrané věty. Pro širší využití aplikace by někteří uživatelé také uvítali podporu českého jazyka při našeptávání textu.

4.4 Příklad tvorby článku

Během uživatelského testování vzniklo deset článků zabývající se různými tématy. V této sekci jeden z těchto článků rozeberu a popíši. Článek jsem vložil bez originálního formátování a tučně jsem vyznačil text, který přidal autor, aby bylo patrné, jaká část článku je vytvořena pomocí dat uložených v aplikaci a kolik textu musel autor do článku přidat sám.

Článek se věnuje jednoduchému řadícímu algoritmu Bubble sort. Podle zadání v dotazníku obsahuje části abstrakt, úvod, vlastní kapitolu a závěr. Autor se v tomto článku chtěl věnovat základnímu popisu a pomalému průběhu tohoto algoritmu.

4.4.1 Abstract

The purpose of this research is to identify **the lacks of bubble sort algorithm**. It describes **the sort algorithm**. This has been done by examining events such as **comparing**. This work demonstrates **slow working of this algorithm**. Results of this work are **recommendation to use other algorithm**.

Abstrakt, spolu se závěrem, je část článku, které museli testovací uživatelé věnovat největší pozornost a tvorba těchto částí jim šla nejpomaleji. Důvodem je, že všechny věty, které aplikace v současné době nabízí pro tyto části článku, jsou parametrizované a uživatelé je musí sami doplnit tak, aby zapadaly do jejich článku. Jak je vidět na vloženém abstraktu, toto doplnění může být od jednoho slova až po dělší text, ale základní kostra byla nabídnuta aplikací.

4.4.2 Introduction

Bubble sort is a simple and well-known sorting algorithm. The bubble sort makes multiple passes through a list **of items**. It compares adjacent items and exchanges those that are out of order. Each pass through the list places the next largest value in its proper place. In essence, each item bubbles up to the location where it belongs. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. It means that for almost sorted array it gives O(n) estimation. **O(n) is a linear complexity.**

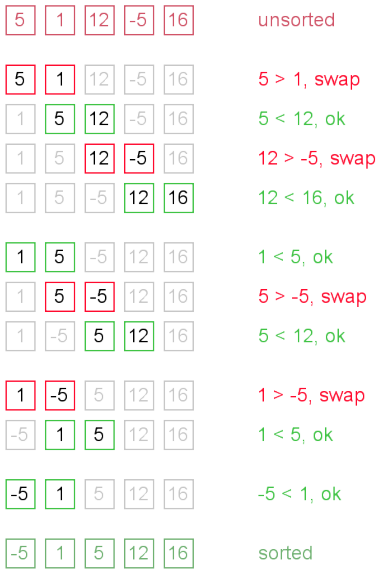
Na začátku této práce jsem se zmínil o struktuře článku a významu úvodu, který má čtenáře uvést do problematiky a přiblížit, čemu se bude věnovat zbytek textu. Úvod tohoto zkušebně vytvořeného článku jednoduše popisuje řadící algoritmus Bubble sort. Autorovi se podařilo vytvořit srozumitelný, souvislý text použitím vět nabídnutých aplikací, pouze s malými opravami.

Jeden z úkolů z testovacího scénáře je přidat do úvodu a vlastní kapitoly alespoň jednu vlastní větu. V úvodu je pro splnění tohoto požadavku vložena věta vysvětlující složitost procházení seřazeného pole.

4.4.3 Bubble Sort details

Bubble sort is one of the most inefficient sorting algorithms. Average and worst case complexity of bubble sort is O(n^2). Also, it makes O(n^2) swaps in the worst case. **O(n^2) is a quadratic complexity.** In bubble sort algorithm,

array is traversed from first element to last element. Bubble sort is stable and adaptive. If current element is greater than the next element, it is swapped. Bubble sort is stable, as two equal elements will never be swapped. **The figure shows the algorithm's progress.**



Obrázek 4.6: Bubble sort

Aby testovací uživatelé vyzkoušeli většinu funkčnosti aplikace, měli za úkol do tvořeného článku přidat jednu vlastní kapitolu a do ní vložit jako přílohu tématický obrázek. Splnění tohoto požadavku ukazuje část vytvořeného článku nazvaná **Bubble Sort details**. V této sekci autor popisuje složitost algoritmu a vysvětluje, co znamená stabilní řadící algortimus a proč je právě Bubble sort stabilní. Autor článku vložil do této kapitoly dvě vlastní věty. První z nich vysvětluje význam složitostni **O(n^2)** a druhá uvádí vložený obrázek. Přiložený obrázek 4.6 reprezentuje průběh algoritmu na poli pěti čísel.

4.4.4 Conclusion

The goal of this work was **the desciption of the buble sort algorithm**. The results suggested that **this algorithm is not usable for many items**. The results showed that **this is a very slow algorithm**. As part of future work, I would like to show **quick sort algorithm**. During the process of desciption I will show **benefits of quick sort**. Finally we **will evaluate this two algorithms**.

Závěr obsahuje, stejně jako abstrakt, všechny věty parametrizované a autor je

musel doplnit podle obsahu článku. Autor využil předpřipravené věty a doplnil, co je obsahem a cílem článku a čemu by se věnoval v dalším článku, který by tvořil o podobném tématu.

4.4.5 Shrnutí

Představený článek má za cíl demonstrovat možnosti aplikace při vytváření vědeckých článků. Autorovi se povedlo splnit všechny úkoly uvedené v testovacím scénáři a vytvořil text o řadícím algoritmu Bubble Sort.

Všem testovacím uživatelům trvalo seznámení s aplikací a vytvoření prvního článku mezi deseti a dvaceti minutami. Většina z nich se pokusila stejným způsobem vytvořit ještě druhý článek a doba, kterou k tomu potřebovali, byla přibližně poloviční.

Kapitola 5

Možnosti dalšího vývoje

Téma mé diplomové práce je velmi obsáhlé. Během implementace jsem musel často vybírat mezi pokročilostí řešení a časovou náročností. Vše, co nelze implementovat najednou v rámci této práce, mohu alespoň navrhnout a doporučit pro případný další vývoj v budoucnu.

V práci je velmi jednoduše řešena manipulace s uloženými články. Do aplikace by mohla být přidána možnost smazat článek, stáhnout článek bez nutnosti otvírat ho v editačním okně. Uživatelé by také mohli ocenit zobrazení náhledu části, popřípadě celého článku před jeho obnovením do editačního okna.

Aplikace v této chvíli nepodporuje tvorbu článku dvěma a více autory najednou. Tato funkce by se dala realizovat umožněním, aby autor sdílet článek s jinými zaregistrovanými uživateli, a uzpůsobit serverovou část aplikace pro nezávislou práci více autorů na jednom článku.

Jednoduše je v práci také řešena funkce vyhledávání informací k článku pomocí internetových vyhledávačů Google a Bing. Uživatelům by nemusely být nabízeny pouze odkazy na stránky, které potencionálně mohou obsahovat zajímavé informace, ale pokoušet se tyto informace přímo vyhledávat a nabízet autorovi článku jako konkrétní text. Ten by bylo možné získávat analýzou odkazů, které jsou nalezeny během vyhledávání.

Výrazné zlepšení by bylo možné přidat do samotného editoru a našeptávače vhodných vět. Uživateli by mohl být po zvolení jedné z vět nabídnut seznam možných nadcházejících vět. Alternativou by byla nabídka celého stromu, kde by uživatel mohl poskládat celý blok textu.

Do editoru TinyMce by mohlo být přidáno více zásuvných modulů pro rozšíření možností formátování textu.

Dalším zásadním vylepšením by bylo automatické rozpoznávání typů vět, které by výrazně urychlilo přidávání nových vět do databáze a tím rozšířilo možnost našeptávání.

Závěr

Cílem této práce bylo navrhnout a implementovat aplikaci, která by na základě předpřipravených textových bloků a kontextu obsahu sestavovaného textu napomáhala svým uživatelům vytvořit vědecký článek.

V práci je nejprve vysvětlena struktura vědeckého článku, kterou výsledná aplikace udržuje. Dále jsou představena řešení zabývající se obdobným tématem.

V další části je vylíčen návrh aplikace a jsou v ní popsány její jednotlivé části a funkce.

Na návrh navazuje realizace aplikace s popisem a ukázkami implementace důležitých funkcí.

Práce je zakončena testováním a vyhodnocením použitelnosti aplikace.

Splnění zadání

Během návrhu a implementace aplikace jsem vycházel ze zadání práce a do řešení jsem zahrnul všechny požadované funkce a vlastnosti. V následujících podsekcích krátce shrnu ke každému bodu zadání, jak byl realizován, nebo která část práce se daným problémem zabývá.

- **výběr předpřipravených částí textu pro jednotlivé části článku** Uživateli jsou nabídnuty samostatné věty, ze kterých si uživatel vybírá, a aplikace se snaží tyto věty pospojovat do smysluplného textu. Tento postup je detailně popsán v sekci 2.6.

- **modifikace vybraných částí textu** V aplikaci jsou dvě možnosti, jak modifikovat textový obsah článku. Na wireframe modelu 2.4 je naznačena editace vět v oblasti aplikace, která slouží pro zobrazení vět nabídnutých aplikací. Další možností je editovat text pomocí wysiwyg editoru TinyMce, ve kterém uživatel vidí kompletní odstavec včetně vložených příloh.

- **nabídka alternativ k formulacím vět s možností doporučování** Systém má začleněný algoritmus, který porovnává rozdíl originálního textu nabídnutého aplikací a uživatelem upravené věty. Díky němu může aplikace získávat nové formulace vět v průběhu tvorby článků uživateli. Takto získané věty jsou pak nabízeny s vyšší prioritou než původní text.

- **jednoduchá interaktivní a iterativní práce** Při uživatelském testování uživatelé kladně hodnotili prostředí aplikace, které jim umožnilo snadno se zorientovat v aplikaci bez využití připravené nápovědy.

  Aplikace je navržena tak, aby uživatel opakováním jednoduchých kroků získal základní kostru odstavce, popřípadě celého článku.

- **jádro aplikace bude realizováno pomocí RESTových služeb, aby bylo možné snadno vyměnit některé části (například doporučování formulací)** Aplikace je navržena jako webový klient komunikující se serverem pomocí http protokolu a RESTového rozhraní. Při jeho návrhu jsem se snažil dodržet standardy RESTové komunikace. Rozbor navrženého rozhraní a komunikace je popsán v sekci 3.2.

- **výstupem aplikace bude návrh textu článku ve formátu LaTex nebo MSWord** Po vytvoření článku mají uživatelé možnost si kompletní text včetně příloh stáhnout do svého počítače v dále editovatelných fomátech *.tex* a *.docx*. Tato funkce je realizována pomocí externího, volně dostupného programu Pandoc a je popsána v sekci 3.7.

Zhodnocení aplikace

Výslednou aplikaci jsem podrobil řadě testů, které ukázaly použitelnost navrženého řešení. Jako největší nedostatek se v současné době jeví velmi omezená zásoba vět, a to i vzhledem k časově náročnému výběru a kategorizaci vět tak, aby byly vhodné pro použití v aplikaci. Tato nevýhoda by se měla zmenšovat s častým využíváním aplikace.

Během návrhu mě napadly některé další funkce, které by mohly vést ke zlepšení práce s aplikací. Několik dobrých nápadů uvedli reální uživatelé, kteří si aplikaci vyzkoušeli v rámci testování. Tyto typy by mohly být velmi užitečné při dalším vývoji aplikace.

Literatura

[1] Oranckay: *A Detailed Review of Instant Article Wizard [online].* [cit. 2017-03-05]. Dostupné z: `http://oranckay.net/a-detailed-review-of-instant-article-wizard-3-0/`

[2] Dr Essay: *SCIgen - An Automatic CS Paper Generator [online].* [cit. 2017-03-05]. Dostupné z: `http://www.dressay.com/`

[3] Ephox: *Full featured web editing. [online].* [cit. 2017-03-28]. Dostupné z: `https://www.tinymce.com/`

[4] Hoogenboom, B. J.; Manske, R. C.: *HOW TO WRITE A SCIENTIFIC ARTICLE [online].* US National Library of Medicine, [cit. 2017-02-27]. Dostupné z: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3474301/`

[5] Bates College: *The Structure, Format, Content, and Style of a Journal-Style Scientific Paper [online].* [cit. 2017-02-27]. Dostupné z: `http://abacus.bates.edu/~ganderso/biology/resources/writing/HTWsections.html`

[6] Mack, C.: How to Write a Good Scientific Paper: Structure and Organization. *SPIE*, 2014.

[7] Lunsford, A.; Connors, R.: *The St. Martin's Handbooke.* Addison-Wesley Publishing Company, 1989.

[8] The University of Sydney: *Typical structure of a paragraph [online].* [cit. 2017-03-13]. Dostupné z: `http://writesite.elearn.usyd.edu.au/m3/m3u2/m3u2s2/m3u2s2_1.htm`

[9] Smalley, R. L.; Ruetten, M. K.: *Refining Composition Skills: Rhetoric and Grammer.* Heinle & Heinle Publishers, třetí vydání, 1996.

[10] Leger, J.: *Instant Article Wizard [online]*. [cit. 2017-03-05]. Dostupné z: http://instantarticlewizard.com/

[11] Stribling, J.; Krohn, M.; Aguayo, D.: *SCIgen - An Automatic CS Paper Generator [online]*. [cit. 2017-03-05]. Dostupné z: https://pdos.csail.mit.edu/archive/scigen/

[12] Nuseibeh, B.; Easterbrook, S.: Requirements Engineering: A Roadmap. *ACM*, 2000.

[13] Čápka, D.: *UML - Use Case Diagram [online]*. [cit. 2017-03-02]. Dostupné z: https://www.itnetwork.cz/navrhove-vzory/uml/uml-use-case-diagram

[14] w3schools.com: *Bootstrap 3 Tutorial [online]*. [cit. 2017-03-15]. Dostupné z: https://www.w3schools.com/bootstrap/

[15] Adaptic, s. r. o.: *WYSIWYG [online]*. [cit. 2017-03-18]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/wysiwyg/

[16] Mencák, T.: *Wireframe - drátěný model - skica webu [online]*. [cit. 2017-03-01]. Dostupné z: https://www.cstechnologies.cz/slovnik-pojmu-wireframe-drateny-model-skica-webu-detail-3325

[17] Foster, N.: *The 20 best wireframe tools [online]*. [cit. 2017-04-05]. Dostupné z: http://www.creativebloq.com/wireframes/top-wireframing-tools-11121302

[18] The HSQL Development Group: *HyperSQL [online]*. [cit. 2017-03-13]. Dostupné z: http://hsqldb.org/

[19] The HSQL Development Group: *Running and Using HyperSQL [online]*. [cit. 2017-03-13]. Dostupné z: http://hsqldb.org/doc/2.0/guide/running-chapt.html

[20] Reese, G.: *Database Programming with JDBC and Java*. O'Reilly Media, druhé vydání, 2000.

[21] Lafon, Y.: *HTTP - Hypertext Transfer Protocol [online]*. World Wide Web Consortium, [cit. 2017-03-15]. Dostupné z: http://www.w3.org/Protocols/

[22] Sandoval, J.: *RESTful Java Web Services*. Packt Publishing, 2009.

[23] Mozilla Corporation: *What is a URL? [online]*. [cit. 2017-03-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL

[24] Friesen, J.: *Java XML and JSON*. Apress, 2016.

[25] Foundation, T. A. S.: *Apache Tomcat 7 User Guide*. Fultus Books, druhé vydání, 2011.

[26] The Apache Software Foundation: *Apache Tomcat [online]*. [cit. 2017-03-15]. Dostupné z: `http://tomcat.apache.org/`

[27] Hunter, J.; Crawford, W.: *Java Servlet Programming*. O'Reilly Media, druhé vydání, 2001.

[28] Perry, B. W.: *Java Servlet & JSP Cookbook*. O'Reilly Media, první vydání, 2014.

[29] JUnit: *JUnit [online]*. [cit. 2017-04-11]. Dostupné z: `http://junit.org/junit4/`

[30] Snozová, M.: *Heuristická analýza [online]*. [cit. 2017-04-05]. Dostupné z: `http://www.inflow.cz/heuristicka-analyza`

[31] Hlava, T.: *Fáze a úrovně provádění test [online]*. [cit. 2017-04-07]. Dostupné z: `http://testovanisoftwaru.cz/tag/unit-testing/`

[32] Mozilla Corporation: *HTTP authentication [online]*. [cit. 2017-04-11]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication`

Příloha **A**

Dotazník k diplomové práci

Dobrý den,

dovoluji si Vás požádat o provedení následujících jednoduchých úkolů a vyplnění dotazníku k mé diplomové práci. Úkoly a dotazník se týkají použitelnosti aplikace, která byla vytvořena v rámci této práce.

1. Vytvořte si nový uživatelský účet, přihlaste se a přejděte na stránku pro tvorbu článku.
2. Zobrazte nápovědu.
3. Zvolte libovolné téma a upřesněte ho výběrem klíčového slova.
4. Doplňte název článku.
5. Ověřte počet vět, které aplikace vygeneruje pro vytvoření abstraktu.
6. Vygenerujte přednastavený počet vět pro abstrakt.
7. U libovolných dvou vět změňte pořadí.
8. Vytvořte krátký článek složený z částí abstrakt, úvod, jedna vlastní kapitola a závěr, který bude odpovídat zvolenému tématu.
9. Do úvodu a vlastní kapitoly přidejte alespoň dvě vlastní věty, které nejsou obsaženy v databázi.
10. Ověřte, jestli systém vyhledal k vloženým větám výsledky, které by mohly pomoci při tvorbě článku.
11. Do vlastní kapitoly přiložte jako přílohu tematický obrázek.
12. Uložte článek jako Word dokument.
13. Odhlaste se.

14. Přihlaste se zpět a zobrazte seznam vašich článků.

---

a) Jak hodnotíte užitečnost nápovědy aplikace (1–nepoužitelná, 10–velmi přínosná)

b) Bylo pro Vás ovládání aplikace intuitivní? (1 - vůbec, 10 - velmi)

c) Odpovídaly nabízené věty zvolenému tématu? (1 - vůbec, 10 - velmi)

d) Poskytuje aplikace smysluplné textové bloky? (1 - vůbec, 10 - velmi)

e) Jak hodnotíte časovou odezvu programu? (1 – velmi pomalé, 10 – velmi rychlé)

f) Byla pro Vás tvorba článku pomocí aplikace jednodušší než tvorba bez této aplikace? (1 - vůbec, 10 - velmi)

g) Doplňte klady aplikace:

h) Doplňte nedostatky aplikace:

i) Doplňte nápady na zlepšení funkčnosti aplikace a návrhu uživatelského rozhraní:

Příloha **B**

Instalační příručka

Aplikace byla vyvíjena na operačním systému Microsoft Windows 10, a proto i instalační příručka popisuje nainstalování aplikace na tento operační systém.

1. Aplikace potřebuje, aby na počítači byla nainstalovaná JAVA jre. Její stažení ve verzi 1.8 je možné ze stránky https://java.com/en/download/. Ověření, že je nainstalovaná správně, je možné příkazem *java -version* v příkazové řádce.

2. Pro správné fungování programu je nutné před jeho spuštěním nainstalovat volně dostupný program pandoc, který je dostupný na adrese http://pandoc.org/index.html. Je důležité, aby po nainstalování byl přístupný z příkazové řádky. To se dá ověřit příkazem *pandoc -v*. Pokud by příkaz nepracoval správně, je nutné přidat pandoc do proměnných prostředí. Cesta, kde je pandoc nainstalován, by měla vypadat přibližně takto C:\Users\{username}\AppData\Local\Pandoc\.

3. Na přiloženém CD je připravená HSQL databáze, kterou zkopírujte na disk. Její spuštění je možné připraveným skriptem hsqldb-2.3.4\hsqldb\runDB.cmd.

4. a V adresáři apache-tomcat-8.5.14\ je připravený aplikační server apache tomcat s již nainstalovanou aplikací. Tento adresář opět zkopírujte na disk. Pro jeho spuštění stačí spustit skript apache-tomcat-8.5.14\bin\startup.bat. Aplikační server pak bude naslouchat na url http://localhost:8080. Uživatelské jméno a heslo do administrátorské konzole je admin/admin.

   b Pokud chcete aplikaci na server nainstalovat sami, stáhněte aplikační server Tomcat z url http://tomcat.apache.org/download-80.cgi. Funkčnost aplikace je vyzkoušena s verzí serveru 8.5.14. Stažený zip soubor stačí rozbalit na disk. Instalace aplikace proběhne zkopírováním souboru dipBE.war do adresáře apache-tomcat-8.5.14\webapps.

Po spustění bude aplikace naslouchat na url http://localhost:8080/dipBE/login.html. Pokud by byl vyžadován přístup k aplikaci z jiného počítače, musely by být upraveny adresy, ke kterým přistupují webové stránky. To se dá provést otevřením souboru dipBE.war v průzkumníku souborů a v adresáři js upravit v souborech app.js, articles.js a login.js hodnotu proměnné var server = "http://localhost:8080/dipBE".

Příloha C

Seznam použitých zkratek

json JavaScript Object Notation

http Hypertext Transfer Protocol

HSQLDB Hyper SQL Database

jre Java Runtime Environment

REST Representational state transfer

WYSIWYG What you see is what you get

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl`
    - `dipBE` ..... zdrojové kódy implementace
    - `sql` ..... skripty pro vytvoření a plnění databáze
    - `libs` ..... adresář s potřebnými knihovnami
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `hsqldb-2.3.4` ..... databáze pro testování práce
- `apache-tomcat-8.5.14` ..... aplikační server Apache Tomcat