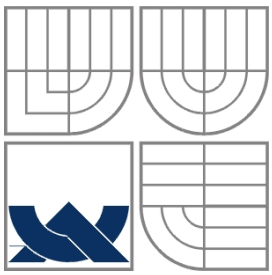
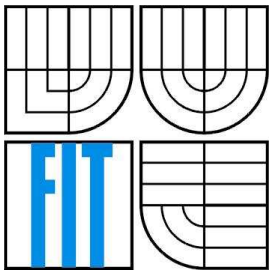


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

UŽIVATELSKÉ ROZHRANÍ PRO SBĚR CITÁTŮ

USER INTERFACE FOR GATHERING QUOTATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — MARTIN VAVERKA
AUTHOR

VEDOUCÍ PRÁCE — ING. ADAM HEROUT, PH.D.
SUPERVISOR

BRNO 2008

Abstrakt

V dnešní době je kladen velký důraz na grafické uživatelského rozhraní. Návrh takového rozhraní obsahuje několik etap. Před samotným návrhem aplikace je důležité si uvědomit, pro koho a za jakým účelem rozhraní vytváříme. Chceme-li, aby se náš návrh stal úspěšným, měli bychom znát a využívat moderních trendů při jeho tvorbě. Je kladen velký důraz na jednoduchost, přehlednost a intuitivnost.

Klíčová slova

Grafické uživatelské rozhraní, návrh GUI, moderní trendy v tvorbě GUI, aplikace pro sběr citátů

Abstract

There is a huge emphasis on graphical user interfaces today. Design of these interfaces compounds of several stages. First thing to do is to find out, whose for is it and what is the purpose of the interface. We should use modern trends to advance our user interface and to be succesfull with it. There is a hudge importance of simplify, intuitiveness and overal lucidity.

Keywords

Graphical user interface, design of GUI, modern trends in GUI creation, aplication for gathering quotations

Citace

Martin Vaverka: Uživatelské rozhraní pro sběr citátů, bakalářská práce, Brno, FIT VUT v Brně, 2008

Uživatelské rozhraní pro sběr citátů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Adama Herouta, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Vaverka
12.05.2008

Poděkování

Chtěl bych poděkovat Ing. Adamu Heroutovi, Ph.D. za jeho odbornou pomoc při vypracovávání této bakalářské práce.

Obsah

1 Úvod

V dnešní době, kdy se práce na počítači stala součástí každodenního života, je kladen velký důraz na návrh uživatelských rozhraní. Přispívá k tomu i fakt, že na trhu existuje velké množství softwaru a uživatel má tak obrovskou možnost volby, který z nich si vybrat. Aby náš návrh uživatelského rozhraní obstál, je nutno jej uživateli co nejvíce přizpůsobit. Musí mu nabídnout jednoduché ovládání a příznivou grafiku. Také je potřeba, aby držel krok s moderními trendy. Používání takto navrhnuté aplikace je pak velmi příjemné.

Tato práce si klade za cíl přiblížit čtenáři tvorbu uživatelského rozhraní. Bude zde zmíněna historie vývoje grafických uživatelských rozhraní a současné trendy při jejich tvorbě. Také zde budou popsány etapy návrhu uživatelského rozhraní pro sběr citátů, výsledná podoba aplikace a řešení jejich stěžejních částí.

2 Grafické uživatelské rozhraní

V této kapitole je popsán vývoj uživatelského rozhraní a následně jsou popsány současné trendy v jeho tvorbě. V historii GUI (zkratka grafického uživatelského rozhraní, která je odvozena z angl. originálu) jsou zmíněny ty nejdůležitější a nejzajímavější události, které vedly k současnému pohledu na uživatelská rozhraní. Tyto události jsou popsány na základě článku Jeremyho Reimera [1].

2.1 Historie

2.1.1 Jak to všechno začalo?

Jak to v historii vývoje počítačů bývá, některé nápady na GUI byly vymyšleny dříve, než vůbec technologie dovolovala sestrojit stroj, ve kterém by se využily. Jedním z prvních lidí, kteří vyjádřili tyto myšlenky, byl Vannevar Bush. Na začátku třicátých let poprvé popsal zařízení nazvané „Memex“, které si představoval jako pult se dvěma dotykovými obrazovkami, klávesnicí a k tomu připojeným snímacím zařízením. Takto si jej tehdy dokázal představit. V té době ale ještě žádný digitální počítač vynalezen nebyl a nebylo tedy možné zjistit, jak by zařízení vlastně fungovalo. Bushovým nápadům tedy nikdo nevěnoval velkou pozornost.

Nicméně kolem roku 1937 začalo několik skupin po celém světě sestavovat digitální počítače. Díky motivaci, kterou vytvořila druhá světová válka, byly vytvořeny programovatelné výpočetní stroje, které se využívaly na řadu věcí (od vypočítávání střel dělostřeleckých tabulí až po prolomování nepřátelských kódů). Aby se počítače staly ještě použitelnějšími, potřebovaly rychlejší přepínací mechanizmus. Ten jim poskytla pozdější výroba vakuových trubic. Nyní byl svět připraven naslouchat Bushovým teoriím. Ten je v roce 1945 znovu zpracoval v článku „*Jak myslíme*“, který byl publikován v měsíčníku Atlantic. Tento článek pak inspiroval mladého Douglase Engelbarta k tomu, aby se pokusil a následně sestavil podobný stroj.

2.1.2 Douglas Engelbart, tvůrce prvního GUI

Douglas Engelbart dokončil své vzdělání v oblasti elektrického inženýrství v roce 1948 a získal práci v ústavu NACA (předchůdce NASA). Jednoho dne si uvědomil, že práce na projektu, z kterého bude mít užitek jen pár lidí, ho moc neuspokojuje. Vždy chtěl dělat na něčem velikém, z čeho by mělo užitek celé lidstvo. Vzpomněl si na Bushovu práci a začal přemýšlet, jak by takový stroj mohl sestrojit. Za války pracoval jako obsluha radaru, díky čemuž získal nápad na sestavení zobrazovacího systému. Ovšem najít někoho, kdo by mu pomohl tento nápad uskutečnit, bylo složité.

V roce 1955 obdržel titul Ph.D. a dostal práci ve Stanfordském výzkumném ústavu, kde získal mnoho patentů za miniaturizaci počítačových součástek. V roce 1962 Douglas publikoval své nápady

v práci „*Zvětšení lidského intelektu*“, v které neviděl počítače jako náhražku lidského intelektu, ale jako pracovní nástroj k jeho zvýšení.

Douglas a jeho postupně vzrůstající se personál pracovali po dlouhá léta na uskutečnění jeho nápadů. Technologii nakonec prezentovali před tisící počítačových profesionálů v roce 1968. Zobrazovací systém byl založen na technologii vektorové grafiky. Díky omezenému paměťovému prostoru dokázal zobrazovat jen velká písmena. Byly vytvořeny tři vstupní zařízení, a to standardní psací klávesnice, pětitlačítková klávesnice a malá obdélníková krabička s třemi tlačítky připojená s počítačem pomocí dlouhého drátu. To byla myš navržená Douglasem a zkonstruovaná jedním z jeho pomocníků. Nikdo neví, kdo ji jako první začal říkat „*myš*“. Nicméně se tohoto názvu již nezbavila. Později se zkoušela nahradit jinými vstupními zařízeními (jako např. dotykový display či světelné pero), ale testováním bylo zjištěno, že použití myši je nejpřirozenější způsob jak hýbat s kurzorem na obrazovce. S vynálezem myši byl vynalezen i její ukazatel, kterým byla šipka. Douglasův tým jej nazval „*brouk*“. Tento název ale dlouho nevydržel.

Douglas a jeho spolupracovníci pokračovali ve vývoji těchto revolučních myšlenek, dokud nebyl ústav v roce 1989 uzavřen.

2.1.3 Xerox a výzkumné centrum Palo Alto

Demonstrace Douglase Engelbarta v roce 1968 ohromila hodně lidí a spoustě z nich otevřela oči. Ti pak viděli možnou budoucnost. Byla to budoucnost, kde lidé spolupracují na elektronických dokumentech, které jsou zobrazeny pomoci obrazovky počítače a které jsou posílány přes síť ostatním uživatelům. Taková budoucnost nevěštila nic dobrého pro společnost, která vydělávala prodáváním kopírek. Tou společností byl Xerox.

Xerox z obavy, že v budoucnu již nebude potřeba jeho výrobků, se rozhodl tuto technologii získat. Založil tak v roce 1970 výzkumné centrum Palo Alto, zkráceně PARC (zkratka je odvozena z angl. názvu Palo Alto Research Center).

V roce 1973 byl sestrojen počítač Alto. Jeho obrazovka měla stejné rozměry a orientaci jako tiskový papír. Rozlišení obrazovky bylo 606 na 808 pixelů. Každý pixel šel nezávisle rozsvítit či zhasnout. Jako vstupní zařízení měl jednu klávesnici a modernější verzi Engebartovi myši, která měla opět tři tlačítka. Ukazatel myši byl zobrazen pomocí bitmapy.

První software napsaný pro Alto obsahoval jen málo grafiky. Grafický procesor slov, zvaný „*Bravo*“, byl vyvinut tak, že dokázal zobrazit text různých fontů a velikostí ve stejném čase. Existoval v něm i grafický bitmapový editor, který pracoval podobně jako dnešní Malování.

Pro různé části programu v tomto softwaru byla vytvořena různá uživatelská rozhraní. Tento přístup se ale výzkumníkům z PARC moc nelíbil, a tak se později rozhodli, že je potřeba u nových aplikací zachovat shodnost uživatelských rozhraní. Vytvořili tedy programovací jazyk a vývojové prostředí Smalltalk, který se stal prvním moderním GUI.

Smalltalk byl pojat jako vývojové prostředí a programovací jazyk. Vývojové prostředí Smalltalku bylo také uživatelským rozhraním, ve kterém běžely jeho programy a představovalo mnoho moderních GUI konceptů. První dostal podobu kolem roku 1974.

Samotná okna Smalltalku byla uvnitř grafických okrajů a stála před šedým pozadím. Každé z nich mělo nad horním okrajem oblast, která obsahovala název, který okno identifikoval a pomocí této oblasti šlo s oknem hýbat. Délka této oblasti nebyla shodná s délkou okna. Její délku udával název. Okna mohla překrývat jiná okna na obrazovce. Označením okna se dané okno přeneslo na vrchol „zásobníku“.

Koncept ikon byl vytvořen ve stejné době. Malé ikonky reprezentovaly programy, se kterými šlo pomocí nich manipulovat. V té samé době bylo také vytvořeno vyskakovací menu, oblasti pro rolování, rádiová tlačítka a dialogové boxy. Kombinací Smalltalku a Alta byl vytvořen osobní počítač s uživatelským rozhraním velmi podobným tomu dnešnímu.

2.1.4 Apple

Apple byla firma, která hledala přísun peněz a nebála se více riskovat. Díky tomu získala mnoho inženýrů z PARC, kteří zde mohli uplatnit své znalosti z práce na Altu a Smalltalku v komerčně využitelných produktech. Tito inženýři byli pro tuto firmu velkým přínosem a při práci na počítači Lisa vytvořili mnoho konceptů GUI.

Počítač nové generace Lisa původně začínal jako tradiční počítač založen na příkazové řádce, který měl obchodní užití. Tento počítač byl ale s přílivem lidí z PARC předělán. Lisa se tak stala počítačem grafickým. Její rozhraní ale nebylo ještě ustálené. Nakonec se tým pracující na Lise rozhodl pro rozhraní založeném na ikonkách, kde každá ikonka znázorňovala dokument či aplikaci.

Při vývoji rozhraní pro Lisu bylo vytvořeno mnoho konceptů, které se dodnes využívají. Například byl vytvořen koncept klávesových zkratek pro často využívané příkazy z menu. Rozhraní Lisy podporovalo dvě akce pro každou ikonku, a to vybrání a spuštění. Jelikož rozhraní Lisy využívalo pouze jednoho tlačítka myši, byl vytvořen koncept dvojitého kliknutí. Dvojité kliknutí se později stalo standardem pro všechna GUI pro spuštění programu.

Zatímco Smalltalk a Xerox Star (upravená komerční verze Alta) používaly ikonky jenom pro reprezentaci souborů jako takových, rozhraní Lisy bylo první, které mělo nápad používat ikonky k reprezentaci všech souborů v souborovém systému, kde byla vytvořena hierarchická struktura adresářů a každý adresář se při otevření zobrazil v novém okně. Nápad označení a přetahování ikonek mezi složkami vznikl ve stejné době podobně jako překreslování pouze oblasti, ve které došlo ke změně, místo překreslování celého okna.

2.1.5 Ostatní GUI osmdesátých let

Apple nebyl sám, který pracoval v té době na uživatelských rozhraních pro osobní počítače. VisiCorp vytvořil vůbec první grafické rozhraní pro počítače IBM. Jmenovalo se VisiOn. Používalo jednobarevný grafický mód a byl spíše textový. Nebyly v něm vůbec použity ikonky a ani nevyužíval proporcionální písmo. Dokonce zde byl použit i vertikální ukazatel místo diagonálního.

VisiOn byl sice neúspěšný, ale inspiroval Billa Gatese k vytvořený produktu, jenž pojmenoval Interface Manager („*Správce rozhraní*“). Později jej přejmenoval populárnějším názvem Windows, který byl představen v roce 1983. Ze začátku byl spíše něco mezi VisiOnem a Microsoft Wordem pro DOS, ale později se stal barevným a obsahoval všechny užitečné prvky GUI. První verze neumožňovala překrývání oken z důvodu matení uživatele. Tento přístup se ale Gatesovi moc nelíbil, a proto všechny ostatní verze již překrývání oken umožňovaly.

Tandy představila první verzi jejich GUI v roce 1984. Nazvali jej DeskMate („*Přítel plocha*“). Ten byl navržen především pro práci s klávesnicí. GUI podporovalo klávesové zkratky, nepodporovalo však překrývání oken. Toto GUI nebylo moc úspěšné.

Později v roce 1985 byl představen GEM, okenní GUI pro DOS. Toto rozhraní bylo velmi podobné GUI počítače Lisa. Bylo zde užito jednoduchého menu, které bylo umístěno u vrcholu obrazovky.

Amiga stejného roku představila své GUI s názvem Workbench („*Pracovní stůl*“). Obsahovalo některé nové myšlenky, jako je schopnost pohybování oken v „*zásobníku*“ a schopnost označit okno, hýbání a pracování s ním bez toho, aby bylo okno automaticky přeneseno dopředu. Toto rozhraní mělo také menu u vrcholu obrazovky. Toto menu bylo skryté a zobrazovalo se zmáčknutím pravého tlačítka myši.

V letech 1987 byl Windows upraven na verzi 2.0. Toto rozhraní využívalo v té době již populární překrývání oken.

V roce 1987 společnost Acorn představila své první GUI. Jmenovalo se Arthut a představilo nový koncept. Tím bylo umísťování běžících programů na lištu umístěnou vespod obrazovky.

NeXTSTEP byl uvolněn v roce 1988. NeXTSTEP představil 3D zobrazení prvků GUI. Byl také prvním, kdo začal používat symbol „X“ k uzavírání oken a představil myšlenku vertikálního rozbalovacího menu, které bylo umístěno v levém horním rohu. Také mohl zobrazovat lišty na jakémkoliv rohu obrazovky. Výchozím umístěním lišty byla pravá strana.

Roku 1988 přišla první grafická verze OS/2. OS/2 byl vytvořen Microsoftem ve spolupráci s IBM za účelem nahrazení DOSu. OS/2 verze 1.0 byl pouze textový, ale verze 1.1 přišla s grafickým uživatelským rozhraním známým jako Presentation Manager („Správce prezentace“). Toto GUI bylo visuelně velmi podobné Windows 2.0.

Před koncem osmdesátých let se začaly objevovat unixové stanice. Tyto stanice běžely na architektuře síťových oken známé jako X, která později našla uplatnění jako GUI pro Linux.

X představilo nový nápad. Stačilo kurzorem myši najet nad okno a to se automaticky aktivovalo. Bylo tedy možné ihned v něm psát. Filosofii rozhraní X bylo ponechání uživatelského rozhraní jednotlivým programům.

2.1.6 GUI devadesátých a pozdějších let

S nástupem devadesátých let spousta platforem přestala být populární. Přeživšími rozhraními byly jen Windows a Macintosh.

Představením Windows verze 3.0 v roce 1990 a verze 3.1 v roce 1992 se zvedla obrovská vlna popularity právě tohoto GUI. Narozdíl od Macintoshe, který měl pořád ještě spoustu nedostatků, Windows byl rázný a měl dobře vypadající ikonky. To vedlo k prodeji miliónů kopií. S příchodem Windows 95 se Microsoft stal hlavním výrobcem GUI na trhu. Windows 95 přišel s novým konceptem položky Start. Přes tuto položku šlo přistoupit ke všem programům a spustit je.

Velká obliba Windows 3 a nedostatečný prodej OS/2 vedl k tomu, že IBM a Microsoft přestali na těchto projektech spolupracovat. IBM si vyvíjela nadále jen OS/2 a Windows přenechal Microsoftu. V roce 1992 představila IBM OS/2 verzi 2.0. Tato verze, díky licenci s Microsoftem, umožňovala, že programy vytvořené pro Windows mohly být spuštěny i na OS/2, a to jak v módu zobrazení přes celou obrazovku, tak i v okně na ploše OS/2.

Apple nechtěl zůstat v pozadí a vyvíjel GUI pro jejich nový operační systém Mac OS X. GUI neslo název Aqua a představilo novou myšlenku tzv. „*double-bufferingu*“. Překreslování oken probíhalo mimo obrazovku a bylo tedy neviditelné. Aqua obsahovala také pár inovací lahodící oku, jako např. rozpínání a stlačování minimalizovaných oken na liště.

2.1.7 Shrnutí

Vývoj uživatelského rozhraní byla dlouhá a strastiplná cesta. Některé týmy byly úspěšnější, jiné méně, ale všechny se zasloužily o to, jakou má dnes GUI podobu a co všechno může nabídnout. Mnoho z konceptů, které dnešní GUI využívá, bylo vytvořeno již v roce 1983 s počítačem Lisa. Následující vývoj pak některé koncepty upravil, jiné přidal.

2.2 Současné trendy tvorby uživatelských rozhraní

V současné době se při návrhu GUI snažíme co nejvíce vyjít vstříc uživateli. Pokoušíme se, aby aplikace námi vytvořené byly tzv. „*šity uživateli namíru*“. Proto, než započneme se samotným návrhem, si musíme uvědomit, pro koho vlastně aplikaci tvoříme a GUI tak přizpůsobit. Dále nás zajímají odpovědi na otázku „*jak*“? V zásadě se dnes řídíme třemi základními pravidly. Snažíme se zejména o jednoduchost, přehlednost a intuitivnost, jak uvádí ve svém článku Jiří Petřek [2]. Podle

Marka Prokopa [3] bychom neměli opomenout ani uživatelský prožitek. Současnými trendy v návrhu GUI tedy jsou:

2.2.1 Jednoduchost

Vytváříme-li aplikaci, určitě je naším záměrem, aby ji mohlo využívat co nejvíce uživatelů. Bude-li aplikace příliš složitá a bude-li muset uživatel prostudovat dlouhý manuál či dokonce projít kurzem, aby mohl náš program používat, pak si buďme jistí, že většina z nich raději zvolí jinou aplikaci. Proto nám při návrhu jde především o to, aby naše navržené GUI bylo natolik jednoduché, že jej uživatel dokáže sám bez potíží ovládat.

2.2.2 Přehlednost

Máme-li na pracovním stole nepořádek, pak naše práce není zdaleka tak efektivní, jak by mohla být. Chvíli trvá, než nalezneme dokument, se kterým chceme pracovat. Další čas nám jistě zabere vytvoření pracovních podmínek k danému úkonu, který chceme provádět. Pokud ale máme krásný přehled, kde co máme a zároveň i čistou pracovní plochu, pak nám stejný úkon trvá podstatně kratší dobu.

To samé platí i v aplikacích. Snažíme se, aby uživatel viděl před sebou vše, co právě potřebuje a zároveň jej nerušily ostatní části programu. Proto se dnes velmi omezují vyskakovací okna. Žádného uživatele jistě nepotěší, když se bude muset proklikat deseti okny, aby mohl provést požadovanou akci. Také čím více oken máme zobrazených, tím menší je přehlednost programu a efektivnost naší práce. Z těchto důvodů jejich použití zůstává jen u částí programu, které se používají ojediněle. Jedná se většinou o okna s různými nastaveními. Části programu, u nichž se předpokládá časté užití, by se měly vyskytovat tak, aby je měl uživatel co nejvíce k dispozici. Hojně se dnes využívají lišty či okna zachytávající se okraje pracovní plochy programu.

Lepší přehlednosti ještě dosáhneme, oddělujeme-li mezi sebou části programu, které spolu nijak nesouvisí.

2.2.3 Intuitivnost

Aplikace by měla být pokud možno co nejvíce intuitivní. Jak toho docílit? Pojmenováváme tlačítka či jiné popisky vhodnými názvy. Také se dnes velmi často používají samovysvětlující ikonky, které nám mnohdy rychleji osvětlí účel tlačítka, než kdybychom se jej snažili popsat textem. K intuitivnosti programu také přispíváme, umisťujeme-li části programu tam, kde bychom je jako uživatelé nejčastěji hledali.

2.2.4 Zaměření na cílovou skupinu

Chceme-li získat kladné ohlasy na naši aplikaci, pak stěžejní etapou při návrhu každého GUI je bezpochyby uvědomění si, kdo ji bude využívat. Toto je nejdůležitější krok. Všechny ostatní kroky v návrhu z něj vycházejí. Snažíme-li se například o to, aby naše aplikace byla co nejjednodušší, pak musíme vědět, kdo s ní bude pracovat. Je určitě velký rozdíl v aplikaci, jejíž GUI bylo vyvíjeno pro vysokoškolské uživatele a aplikací vytvořenou pro seniory. Proto každá naše otázka při návrhu musí být zodpovězena právě z pohledu budoucího uživatele.

2.2.5 Co největší usnadnění práce s programem

Dalším moderním trendem je usnadňování práce uživateli. Snažíme se podle jeho minulé práce zjistit, jaká bude ta budoucí. Vytvářejí se tzv. *„našeptávání“*. Píše-li např. text, který již dříve psal, dostává možnost doplnit zbytek tohoto textu. Nabízíme mu poslední otevřené projekty v naší aplikaci, pamatujeme si poslední otevřenou složku, v kontextovém menu řadíme položky podle nejčastějších užití atd. Vždy se snažíme na prvním místě uživateli nabídnou to, co s největší pravděpodobností bude chtít použít.

2.2.6 Možnost nastavení

Poskytneme-li uživateli možnost si nastavit program k obrazu svému, uživatel nám bude jedině vděčný a my tím můžeme jedině získat. Každý člověk má trochu odlišné vnímání. Někomu se může zdát rozvržení aplikace či použití barev v ní vhodné, jinému zase ne. Možnost nastavení je důležitá. Třeba jen volba barev nám může z nepříliš zajímavé aplikace vytvořit docela atraktivní.

2.2.7 Důraz na uživatelský prožitek

V neposlední řadě nám jde také o to, aby byla práce s naší aplikací příjemná. Grafická stránka je dnes velmi důležitá a klade se na ni velký důraz. Vhodně navržené GUI přitahuje pozornost uživatelů. Naopak nevhodně navržené ji ztrácí. V dnešní době máme spoustu možností jakou aplikaci použít. Například neexistuje jen jeden program na editaci textu. Je jich nepřeberné množství. A pro uživatele bude hlavním kritériem pro volbu právě to, jak aplikace vypadá, jak na něj její GUI působí. Jak program pracuje on nevidí. Ovšem jak program vypadá, může zhodnotit ihned.

Jaké prvky mohou uživatele zaujmout? Například se dnes používá postupné vyjíždění postranních lišt. Případně zvětšování ikonek pod kurzorem myši. Lze i vhodně využívat průhlednosti na části či celou aplikaci. Zkrátka prvků jak oživit námi navržené GUI může být mnoho. Je jen na nás jak je dokážeme zkombinovat.

3 Návrh GUI

Cílem práce je vytvoření uživatelského rozhraní pro sběr citátů. Toto rozhraní musí splňovat určité požadavky. Následující podkapitoly obsahují formulaci těchto požadavků. Sled těchto podkapitol je vytvořen podle etap, které byly při návrhu uživatelského rozhraní řešeny.

3.1 Požadavky na GUI

Před každým návrhem GUI je vhodné si definovat, co od aplikace očekáváme. Aplikace by měla být co nejpoužitelnější. Proto by měl její návrh využívat moderních trendů a přizpůsobit se účelu aplikace. Od aplikace pro sběr citátů očekáváme snadné přidávání citátů, jejich mazání, editaci a vyhledávání. Databáze by měla být zobrazena vhodným a přehledným způsobem. Dále by uživatel měl mít možnost si k citátům psát poznámky či si nechat zkontrolovat pravopis u ukládaných citátů.

3.2 Předpokládaný uživatel

Každá aplikace je vytvořena nejen za jistým účelem, ale také pro určitou skupinu lidí. Při návrhu uživatelského rozhraní je nutno tuto skutečnost zohlednit. Dalším krokem je tedy definování uživatele, pro kterého aplikaci vytváříme.

Předpokládaným uživatelem aplikace pro sběr citátů je člověk, jenž si rád čte knížky a čas od času si z nich vypisuje oblíbené pasáže. Při své zálibě má již několik desítek citátů zaznamenaných ve svém bloku. Poté se mu dostane do rukou tato aplikace. Aplikace by tedy měla počítat s tím, že uživatel může chtít přidávat několik citátů z téže knihy naráz a nebude chtít pořád vyplňovat údaje o knize. Také by měla předpokládat, že uživatel nemusí mít znalost o všech údajích, které se uvádějí při bibliografické citaci a že si zaznamenal mnohdy pouze název knihy a jméno autora u svých citátů. Taková by mohla být většina uživatelů chtějících tuto aplikaci využít a aplikace by tomu měla být přizpůsobena.

3.3 Návrh

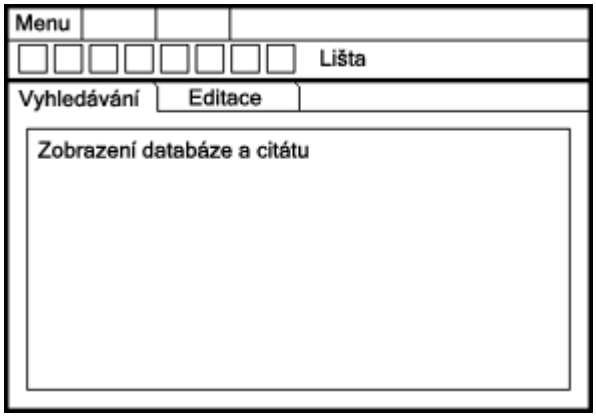
Poté, co jsou vytvořeny požadavky na GUI, se můžeme pustit do samotného návrhu. Toto je nejdůležitější etapa při vytváření každého GUI. Dobrý návrh nám ušetří spoustu času při vytváření aplikace, naopak špatný návrh povede k tomu, že při vytváření aplikace se budeme muset neustále vracet do etapy návrhu a návrh upravovat.

Bylo vytvořeno několik návrhů na rozvržení aplikace. První je zobrazen na obrázku 2.1. Podle tohoto návrhu je program rozložen do dvou oblastí. V jedné oblasti se zobrazují citáty a v druhé

se pracuje s databází. Oblast pro práci s databází je rozdělena do několika záložek. Jedna záložka zobrazuje databázi, druhá poskytuje vyhledávání v databázi a třetí slouží k editaci a přidávání citátů.

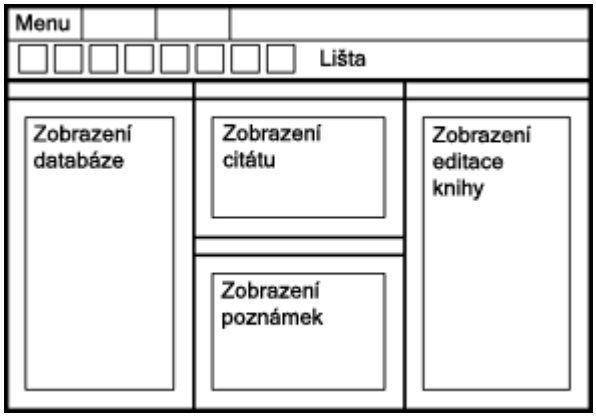
Druhý návrh použil jiný přístup pro práci s databází. Celá aplikace je rozdělena do dvou záložek. První slouží k práci s databází. V té je možno citáty zobrazovat, vyhledávat, editovat a mazat. Druhá je pak určena jen pro přidávání. Aplikace je tak rozdělena podle dvou odlišných akcí, které by uživatel mohl chtít provádět po spuštění aplikace. Tento přístup vychází z úvahy, že uživatel chce buď citáty přidávat nebo zobrazovat. Aplikace mu tedy zobrazí jen informace, které v danou chvíli potřebuje znát. Tento návrh vidíme na obrázku 2.2.

Obrázek 2.1: První návrh GUI



Obrázek 2.2: Druhý návrh GUI

Posledním návrhem byl návrh, který je zobrazen na obrázku 2.3. Podle tohoto návrhu jsou části aplikace rozděleny do několika oken, které si uživatel může v rámci aplikace rozmístit kam chce. Důvodem pro vytvoření tohoto návrhu je větší svoboda uživatele. Tento návrh byl nakonec použit a je blíže popsán v kapitole 3 - Popis GUI.



Obrázek 2.3: Výsledný návrh GUI

Po vytvoření grafické části rozhraní se přechází k návrhu funkčnosti aplikace. Návrh funkčnosti je z celého návrhu nejobtížnější a zabírá nejvíce času. V této etapě je řešena nejen úloha

jednotlivých prvků GUI, ale také spolupráce mezi nimi. Jak byly vytvořeny stěžejní části aplikace je popsáno v kapitole 4 – Realizace aplikace.

3.4 Výběr vývojového prostředí

Jakmile je vytvořen návrh, je třeba najít vhodné vývojové prostředí, které nám umožní aplikaci vytvořit. Dnes se často používá na vytváření GUI pro aplikace tzv. WYSIWYG editor (zkratka pochází z angl. názvu What You See Is What You Get), který umožňuje programátorovi GUI snadno rozvrhnout. Proto byl na vytvoření aplikace pro sběr citátů použit WYSIWYG editor Microsoft Visual Studia 2005. Programovacím jazykem se stal C#.

4 Popis GUI

Tato kapitola popisuje výsledný návrh aplikace pro sběr citátů. Jsou v ní také popsány důvody, které k tomuto řešení vedly.

4.1 Různé druhy pohledů v aplikaci

V aplikaci existuje několik druhů zobrazení pracovních oken. Každé uspořádání bere v úvahu jistou činnost uživatele. Nazveme-li zobrazení oken jako pohled, pak v aplikaci existuje univerzální pohled, dva pohledy pro zobrazování citátů a jeden pro vkládání. Univerzální pohled byl vytvořen pro editaci citátů. V tomto pohledu jsou zobrazena všechna okna a editace se tak stává jednoduší. Chce-li si uživatel prohlédnout citáty a zvolí-li jeden z pohledů pro zobrazení citátů, pak se před ním skryje okno pro vkládání a případně i okno s poznámkami. Nedochází tak k rušivým elementům. Obdobně to je i v pohledu pro vkládání, kde se nezobrazuje okno na vyhledávání. Uživatel má také k dispozici si jeden vlastní pohled vytvořit. Může si nadefinovat, jaká okna se zobrazí a kde je chce mít.

4.2 Okenní systém

Uživatel k vytvoření vlastního pohledu využije vlastností okenního systému. Ten byl vytvořen za účelem poskytnout uživateli co největší svobodu. Díky němu si tak může přizpůsobit aplikaci k obrazu svému. Hýbat s okny lze uvnitř pracovní plochy aplikace a lze je k okrajím této plochy přichytávat. Při změně velikosti okna aplikace se pak přichycená okna drží daného okraje a pohybují se s ním. Uživatel si také může nastavit velikost těchto oken.

Každé okno obsahuje kontextové menu, pomocí kterého je možno okno vypnout či přichytit k okraji pracovní plochy aplikace. Využije-li uživatel možnost přichytit okno pomocí kontextového menu, dojde k jeho umístění k danému okraji a okno si následně udržuje velikost okraje. Takto přichycené okno zmenší pracovní plochu o svou velikost a ostatní okna jej již nemohou překrývat. Jelikož se nyní pracovní plochy neúčastní, lze k němu ostatní okna přichytávat. Pokud se toto okno vypne, pracovní plocha se zase zvětší.

Okna lze uzamknout. Jsou-li okna uzamčena, s okny nelze hýbat ani měnit jejich velikost. Tato možnost byla vytvořena k tomu, aby nedocházelo k nechtěnému rozházení vlastního rozmístění oken. Použije-li uživatel jeden z přednastavených pohledů, okna jsou automaticky uzamčena.

Okna lze kdykoliv vypnout či zapnout. U přednastavených pohledů lze vypínat a zapínat jen ta okna, která se pohledu zúčastňují. Dojde-li k vypnutí takového okna, pohled se tomu přizpůsobí. Volný prostor vyplní okno s citátem a okno s poznámkami, jsou-li zapnuta. Při vypnutí ve vlastním pohledu dojde pouze ke zvětšení pracovní plochy.

4.3 Práce s databází

Na zobrazení databáze byla použita stromová struktura. Stromová struktura je velmi přehledná a poskytuje možnost nechat si zobrazit pouze informace, které chceme znát (např. citáty od jednoho autora). V aplikaci byly vytvořeny dva pohledy na databázi.

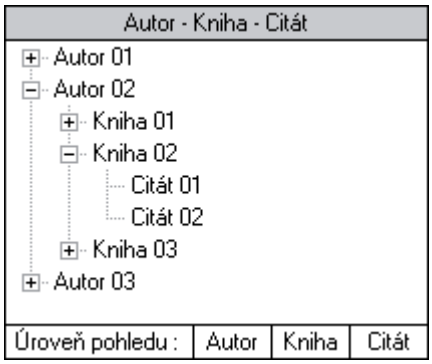
Jedním pohledem je zobrazení typu Autor-Kniha-Citát. Při tomto zobrazení dosahuje stromová struktura tří úrovní. Na první úrovni jsou uzly se jmény autorů. Na druhé úrovni jsou názvy knih daného autora. Na poslední je pak seznam citátů z každé knihy. Tento typ pohledu kopíruje strukturu ukládání dat do databáze. Můžeme jej vidět na obrázku 3.1.

S uzly různých úrovní lze pomocí kontextového menu provádět různé operace. Na úrovni autorů lze k autorovi přidat knihu, autora editovat či smazat. Obdobné operace lze provádět na úrovni knih, rozdíl je pouze v přidání citátu místo knihy. Aktivuje-li se uzel na úrovni citátů, dojde k zobrazení daného citátu. Citát je pak možno editovat či smazat.

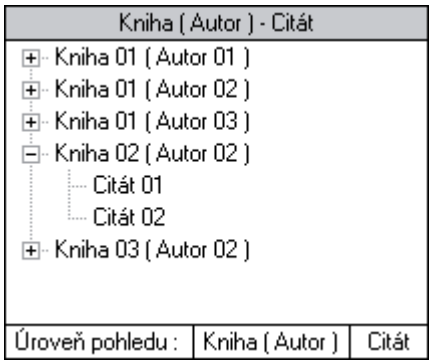
Uživatel může libovolně procházet mezi danými úrovněmi a provádět dané operace. Uživatel by ale mohl také chtít vidět všechny knihy v databázi nebo zobrazit všechny citáty. Aby nemusel procházet všemi uzly a následně je rozbalovat, byla vytvořena tlačítka na zobrazení dané úrovně. V případě zobrazení typu Autor-Kniha-Citát jsou tyto tlačítka tři a zobrazují tyto úrovně.

Druhým typem pohledu je zobrazení Kniha (Autor) – Citát a je zobrazen na obrázku 3.2. Při tomto typu má stromová struktura jen dvě úrovně. Na první úrovni jsou názvy knih, u kterých je v závorce uveden jejich autor. Další úroveň pak patří citátům. Operace prováděné nad těmito uzly jsou obdobné jako u typu Autor - Kniha - Citát. I v tomto pohledu se dá přepínat mezi různými úrovněmi uzlů.

Položky ve stromové struktuře jsou řazeny podle abecedy. První typ byl vytvořen jako standardní přístup k citátům od autora přes knihu. Druhý typ pak slouží těm uživatelům, kteří chtějí mít abecedně seřazeny knihy a pro které není jméno autora prioritní pro řazení citátů.



Obrázek 3.1: Zobrazení 1. typu



Obrázek 3.2: Zobrazení 2. typu

4.4 Přidávání nových citátů

Pro přidávání citátů lze využít buď univerzální pohled nebo pohled pro vkládání. Aby byl citát úspěšně vložen do databáze, musí uživatel vyplnit povinné údaje. Mezi povinné údaje patří název knihy, jméno autora a označení citátu. V přidávaném citátu či poznámkách má uživatel možnost si určitý text odlišit. Zvýraznit text lze třemi způsoby. Text může být psán tučným písmem, podtrženým nebo kurzívou. Po přidání nového citátu zůstanou uživateli údaje o knize předvyplněné. Tyto údaje se nesmažou z toho důvodu, kdyby uživatel chtěl zadávat velké množství citátů z jedné knihy. Nebude tak muset stále vyplňovat ty samé údaje. Efektivnost vytváření databáze se tak zvýší. Nechce-li této možnosti využít, údaje jednoduše vymaže tlačítkem „*Zrušit*“.

4.5 Zobrazování citátů

Při použití této aplikace se předpokládá, že uživatel stráví nejvíce času čtením citátů. Měla by se mu tedy tato činnost co nejvíce zpříjemnit. Proto byla vytvořena možnost, aby si uživatel mohl zvolit barvu a typ písma, kterým budou citáty prezentovány. Změna zobrazení se projeví u všech citátů.

Další možností, kterou uživatel získal je zvětšení písma. Tato možnost byla vytvořena především pro hůře vidící uživatele. Zvětšení písma se provádí pomocí pohybu myši. Aktivování zvětšování či zmenšování písma se děje přes pravé tlačítko myši. Pohybem po ose Y se mění velikost písma. Mění-li kurzor pozici směrem vzhůru, logicky se pro uživatele písmo zvětšuje. Provedením opačného pohybu dochází k zmenšování, a to tak dlouho, dokud se velikost písma nevrátí na původní hodnotu. Základní velikost písma je 10. Úroveň přiblížení písma se bere jako vlastnost zobrazení citátů. Stačí si tedy jeden citát přiblížit a změna se projeví i u ostatních.

4.6 Kontrola pravopisu

V citátech či poznámkách lze zkontrolovat pravopis. Tato možnost byla vytvořena pro případ, že by si uživatel chtěl ověřit, zda se při psaní citátu někde nepřepsal. U slov, jenž nejsou správně napsaná, se změní barva písma na červenou. Kontrola pravopisu proběhne vždy jen na vyžádání uživatele.

Do databáze slov pro kontrolu pravopisu je možné přidávat nová slova. Existují dvě možnosti, jak může uživatel nová slova přidávat. Jednou z nich je přidávání slov uživatelem, druhou je analýza vloženého textu.

Chce-li uživatel přidávat nová slova osobně, využije k tomu vkládací pole. Jednotlivá slova se od sebe oddělují jedním z členících znamének. Po potvrzení se uživateli vypíše seznam slov,

který má být přidán do databáze. V této fázi se ještě může uživatel rozmyslet a ze seznamu některá slova odstranit. Následně potvrdí přidání těchto slov do databáze.

Bude-li chtít uživatel využít analýzy textu, stačí když do vkládacího pole vloží zvolený text a nechá jej zanalyzovat. Slova, která databáze nezná, se mu v textu zvýrazní a jejich seznam se vypíše. Tento seznam může uživatel ještě upravit a následně přidat do databáze. Tento přístup byl vytvořen k rychlému zvyšování korektních slov databáze. Vkládáme-li text, v němž existují podle nás či jiných programů jen správně psaná slova, pak se databáze snadno a rychle těmto slovům naučí.

Uživatel by se mohl chtít podívat, jaká slova databáze zná. Byla tedy tato informativní možnost vytvořena. Uživatel si může zobrazit seznam slov začínající daným písmenem. Kromě tohoto výčtu se mu zobrazí i počet těchto slov.

5 Realizace aplikace

V této kapitole je popsán princip řešení stěžejních částí aplikace. Je zde uveden okenní systém, vysvětlena práce s databází citátů a popsána kontrola pravopisu. Také jsou v této kapitole uvedena řešení, která neměla velký úspěch.

5.1 Okenní systém

5.1.1 Reprezentace okna

Aby měl uživatel větší svobodu, obsahuje aplikace okenní systém. Díky němu má uživatel možnost si vytvořit vlastní uspořádání pracovní plochy. V aplikaci existují celkem čtyři okna (jedno pro zobrazení databáze citátů, jedno pro vkládání údajů o knize, jedno zobrazující citát a jedno zobrazující poznámky). Každé okno je reprezentováno pomocí dvou panelů. Jeden panel znázorňuje okno jako takové, druhý tvoří záhlaví. Prvky okna jsou umístěny na hlavním panelu. Záhlaví slouží k manipulaci s oknem.

5.1.2 Přichytávání oken k okrajům pracovní plochy

U okrajů pracovní plochy byla vytvořena oblast, která určuje, zda má být okno přichyceno či nikoliv. Při pohybu s oknem se pak ověřuje, zda se okno nepohybuje uvnitř této oblasti. Zasáhne-li do této oblasti, pak je jeho pozice upravena. Tím dochází k požadovanému přichytávání k okraji. Je také nutné zajistit, aby se dané okno u okraje udrželo. Naštěstí objekt třídy *Panel* má vlastnost „*Anchor*“, která určuje, podle kterého okraje bude jeho vzdálenost stejná. Stačí tedy jen nastavit, kterých okrajů se přichytávání týká.

5.1.3 Přichycení okna k okraji aplikace

Přichytávání oken k okraji aplikace je řešeno pomocí vlastnosti objektů třídy *Panel*, jenž je pojmenována názvem „*Dock*“. Přichycení je pak k danému okraji velmi jednoduché, stačí změnit její hodnotu.

5.1.4 Pracovní plocha aplikace

Pracovní plocha aplikace je oblast, ve které lze u oken měnit jejich velikost a uvnitř které se mohou okna pohybovat. Ke změně velikosti této plochy dochází při změně velikosti celé aplikace, je-li nějaké okno vypnuto či přichyceno k okraji aplikace nebo je-li u takového okna měněna jeho velikost.

Velikost pracovní plochy se získá odečtením šířky či výšky přichycených oken. Následně je zjišťována pozice všech volných oken a nalézají-li se mimo tuto oblast, jejich pozice je upravena.

5.1.5 Různá rozmístění pracovních oken

Aplikace má čtyři přednastavená rozmístění pracovních oken. Při zobrazení každého z nich je u všech oken změněna jejich pozice, výška a šířka. Dále se nastavuje, zda okno má být přichyceno a vzhledem ke kterému okraji se upravuje jejich pozice. Použije-li uživatel jedno z těchto rozmístění, pak nelze s okny hýbat ani měnit jejich přichycení u okraje aplikace. Jediné co uživatel může udělat s okny, která jsou v daném rozmístění zobrazena, je zapínat a vypínat. Je-li jedno z těchto oken vypnuto, dochází k následnému upravení daného rozmístění. Prázdná oblast je vyplněna pomocí okna s citátem a okna s poznámkami. Není-li zapnuto ani jedno z těchto dvou oken, oblast zůstane nevyplněná.

5.1.6 Problémy okenního systému

Využijeme-li WYSIWYG editor, má to tu výhodu, že se nemusíme starat o překreslování. Stačí umístit jednotlivé prvky GUI a nastavit, podle kterého okraje se jejich pozice bude určovat. O překreslování se editor postará sám. Nevýhodou tohoto přístupu u okenního systému je, že při změně velikosti vnitřního okna dochází k pomalému překreslování jednotlivých částí.

5.2 Databáze citátů

5.2.1 Struktura databáze

Navrhujeme-li strukturu databáze, musíme především vědět, jaká data v ní chceme ukládat a jaké operace budeme chtít nad těmito daty provádět. Struktura databáze byla navržena s ohledem k těmto skutečnostem tak, aby pozdější práce s ní byla co nejefektivnější.

V případě databáze citátů musíme ukládat kromě samotných citátů minimálně i údaje, které jsou nezbytné pro jednoznačnou identifikaci každého z nich. Celkem vhodné je použít kombinaci informací o autorovi, knize a samotném citátu. Právě daná kombinace nám přináší velké výhody, např. kdybychom pojmenovávali citáty podle stran, na nichž se v dané knize nacházejí, můžeme bez problémů použít v jiné knize to samé pojmenování.

Kromě informací, které jsou nutné ukládat, existují i informace vhodné k uložení. Těmi budou bezpochyby údaje o knize. Tato všechna data by nám jistě stačila. Nicméně uživatel může být jiného názoru. Chtěl by si třeba ukládat jaké byly jeho první dojmy nebo kde by tento citát chtěl použít. Dáme tedy uživateli možnost psaní poznámek k citátům, a tím postihneme všechno ostatní, co by ještě uživatel mohl chtít mít zaznamenáno.

Nyní, když máme základní výčet toho, co všechno ukládat, nás začne zajímat jak. Ke strukturovanému ukládání dat je dnes hojně využíváno značkovacího jazyka XML (zkratka je odvozena z angl. názvu Extended Markup Language). Využijeme-li tohoto formátu a pojmenujeme-li vhodně názvy elementů a atributů, databáze se stane snadno čitelná nejen pro naši aplikaci, ale i pro nás samotné. Náročnější uživatel se pak snadno v databázi vyzná a bude ji moci editovat i bez pomoci naší aplikace. Ukázka databáze je na obrázku 4.1.

Obrázek 4.1: Databáze citátů

5.2.2 Systém práce s databází

Pro práci s databází citátů je vytvořena třída *XmlList*. Po zavolání konstruktoru se vytvoří hlavička xml dokumentu a prázdný seznam citátů. Můžeme tedy ihned po vytvoření instance třídy pracovat s databází, aniž bychom byli nuceni k jejímu načtení ze souboru. Metody této třídy zajišťují zobrazení záznamů, jejich vytváření, mazání a editaci. Samotnou databázi pak reprezentuje instance třídy *XmlDocument*. Použití této reprezentace má tu výhodu, že se mezi autory, knihami či citáty pohybujeme jako v poli. Další výhodou je, že nemusíme databázi složitě vytvářet. O načtení či uložení se starají metody *Save* a *Load* náležící právě této třídě. Databáze je ukládána v xml verzi 1.0 s kódováním UTF-8.

5.2.3 Načítání a ukládání citátů

Při přidávání citátu se vždy ověřuje, zda databáze již informace o autorovi či knize neobsahuje. Pokud se tyto informace v databázi nenacházejí, vytvoří se. Existují-li, pak se na ně naváže. Přidáváme-li tedy více citátů z jedné knihy, informace o knize či autorovi je pouze jedna. Nesouhlasí-li údaje o knize v databázi s údaji, které vyplnil uživatel, pak uživatel musí určit, která data jsou pravdivá a ta budou uložena.

Načítání citátů je pak velmi jednoduché, stačí zjistit index autora, knihy a citátu a následně vypsat dané hodnoty atributů (v případě citátu a poznámky obsahy elementů).

5.2.4 Korekce mezi RTF a značkovacím jazykem databáze

Uživatel má možnost si různým způsobem zvýraznit části citátů. Proto pro vkládání citátů či poznámek je v aplikaci použit objekt *RichTextBox*. Ten nám dokáže dát text ve formátu RTF (zkratka pochází z angl. názvu Richt Text Format). RTF je značkovací jazyk určen k formátování textu. Získáme tak informace o typu písma, velikosti textu, jeho barvě atd. Z tohoto ohromného množství dat se využívají jen informace o tom, kde je použito tučné písmo, kde podtržené a kde kurzíva. Je tedy zbytečné ukládat text ve formátu RTF. Z toho důvodu se provádí konverze mezi RTF a značkovacím jazykem databáze.

Značkovací jazyk databáze používá značky, které jsou stejné jako tagy pro odřádkování, tučné či podtržené písmo a kurzívu v HTML (zkratka pochází z angl. názvu Hypertext Markup Language). Samotný soubor s databází se tak stane snadno čitelným a editovatelným.

Při konverzi z RTF na značkovací jazyk nejprve dochází k odstranění hlavičky. Hlavička obsahuje informace o verzi RTF, kódové stránce, použitých fontech a barvách atd. Tyto údaje není potřeba ukládat. Následně se provede konverze řídících slov RTF na značky jazyka databáze a vytvoří se interpunkce.

Při konverzi ze značkovacího jazyka databáze na RTF postupujeme opačně. Vytvoříme hlavičku a přepíšeme značky na řídící slova.

5.2.5 Editace záznamů

V aplikaci je možno editovat údaje o autorovi či knize. Chceme-li upravit záznam citátu, dojde k jeho odstranění a k vytvoření nového. Editujeme-li autora, vytvoří se nový záznam o autorovi, na něj se naváží záznamy o knihách, které byly vázány ke starému záznamu, který je následně odstraněn. Máme-li již záznam o autorovi v databázi, pak se záznamy o knihách k němu přidají. V případě editace údajů o knihách se provádí obdobná operace.

5.2.6 Vyhledávání v databázi

Můžeme použít filtr a nechat si tak zobrazit jen část databáze. Filtr lze použít na všechny údaje o citátu nebo jen na údaje o autorovi, knize či obsah samotného citátu. Zjistí se tedy, kde všude se má daný text vyskytovat a pokud je nalezen, pak se záznam zobrazí uživateli v seznamu nalezených. Filtr je možno použít na zmáčknutí nebo inkrementálně.

5.3 Kontrola pravopisu

5.3.1 Struktura databáze uložených slov

Struktura databáze slov pro kontrolu pravopisu byla vytvořena s ohledem na množství dat, která v ní budou uložena a na operace prováděné nad těmito daty. Databáze je uložena ve formátu xml. Slova uložená v databázi jsou rozdělena podle délky a dále pak podle počátečních písmen. Jednotlivá slova stejné délky a shodného počátečního písmena jsou mezi sebou oddělena znakem \. Takováto struktura ukládání je kompromisem pro rychlé vyhledávání slov a pro malou velikost souboru s databází. Ukázka databáze je na obrázku 4.2.

Základní databáze obsahuje 161 370 slov. Uživatel může přidávat do databáze svoje vlastní slova. Ta se pak ukládají do databáze jako slova neznámé délky. Tím jsou od původní databáze oddělena. Databáze se tak stane přehlednější a navíc je možno kdykoliv uživatelskou část vymazat.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <List>
  - <Length value="2">
      <A>\ač\ad\ah\ai\aj\ak\ap\aš\ať\au\až</A>
      <B>\by</B>
      <C>\či\čj\čl\co</C>
      .
      .
      .
    </Length>
    .
    .
    .
</List>
```

Obrázek 4.2: Databáze slov pro kontrolu pravopisu

5.3.2 Systém práce s databází slov

Pro práci s databází slov pro kontrolu pravopisu je vytvořena třída *OrthographyDatabase*. Po spuštění aplikace se program pomocí této třídy pokusí databázi načíst. Nepovede-li se mu to, pak se pracuje jakoby s prázdnou databází. Chybí-li tedy uživateli databáze slov pro kontrolu pravopisu, může si ji sám vytvořit.

5.3.3 Kontrola pravopisu

Text, u něhož chceme zkontrolovat pravopis, se rozloží na jednotlivá slova. Vytvoří se tak seznam slov, u nichž budeme provádět kontrolu pravopisu. Seznam se následně upraví tak, aby se v něm vyskytovalo každé slovo jen jednou. Poté se zjišťuje, zda se dané slovo označí jako neznámé či nikoliv. V databázi jsou uložena všechna slova malými písmeny. Proto, než se začne se slovem pracovat, musí se vyjádřit právě pomocí malých písmen. Také se zjišťuje, zda se nejedná jen

o písmeno či číslo. Pokud ano, tak se pravopis kontrolovat nemusí, neboť písmena a čísla jsou vždy napsaná správně. V ostatních případech se prohledá databáze. Slovo se porovnává jen se slovy, která mají stejnou délku se shodnými počátečními písmeny. Slovo neprošlé kontrolou pravopisu se zvýrazní. Zvýraznění je provedeno změnou barvy písma.

5.3.4 Přidávání nových slov do databáze

Než se do databáze přidá nové slovo, ověřuje se, zda již v databázi neexistuje. Dále se do databáze neukládají čísla a samotná písmena. Potenciální slova, která se mají uložit do databáze, lze nalézt dvěma způsoby. Buď je může uživatel sám zadat nebo má možnost nechat zanalyzovat vložený text a aplikace mu pak sama nabídne seznam neznámých slov.

5.3.5 Přidávání slov z vkládacího pole

Za slovo se bere jakýkoliv text, který je oddělen jedním z členících znamének. Poté, co se jednotlivá slova od sebe oddělí, se vytvoří seznam potenciálních slov. Před přidáním dalšího slova do seznamu se ověřuje, zda již v seznamu dané slovo neexistuje. Uživatel má pak možnost ještě seznam upravit, než nakonec přidá slova do databáze.

5.3.6 Přidávání slov pomocí analýzy textu

Před analýzou textu se vyčistí seznam potencionálních slov. Tím docílíme větší přehlednosti pro uživatele, jemuž se tak zobrazí pouze výsledek analýzy. Na text vložený do vkládacího pole se použije kontrola pravopisu. Slova neznámá pro databázi se tak zvýrazní a zároveň se přidají do seznamu potenciálních slov. Uživatel si pak může prohlédnout slova, která se mají přidat do databáze, přímo v textu, případně se podívat na jejich výčet.

5.4 Řešení, která se neosvědčila

5.4.1 Původní návrh aplikace

Původně byla databáze navržena tak, aby citáty v ní uložené se zobrazovaly v takovém tvaru, ve kterém je bylo možné ihned použít. Kromě samotných citátů se tedy zobrazovala i hlavička citace podle daných norem. Nevýhodou tohoto přístupu bylo, že chtěl-li uživatel mít hlavičku správně, musel vyplnit všechny údaje. Chtít po uživateli, aby vždy vyplnil všechny údaje, ovšem není vhodné. Navíc je všechny ani nemusí znát. Proto bylo od tohoto návrhu upuštěno. V současné aplikaci jsou uživateli nabídnuty údaje, které by bylo vhodné vyplnit, ale není k tomu nucen. Pokud chce mít citaci podle dané normy, má možnost si tvar citace uložit do poznámek.

5.4.2 Nevhodná reprezentace databáze v aplikaci

U současné aplikace je databáze ukládána ve formátu XML. Tak tomu bylo vždy. Ovšem nebyla vždy reprezentována v aplikaci pomocí třídy *XmlDocument*. Původně ke čtení ze souboru, ukládání i samotné reprezentaci databáze byla využívána třída *DataSet*. Tato třída byla celkem vhodná pro reprezentaci databáze. Šlo pomocí ní provádět všechny operace, které byly potřeba. Nicméně měla jednu obrovskou nevýhodu. Celkem složitě se v ní vytvářela hierarchická struktura XML. Aby databáze byla co nejčitelnější, hierarchická struktura byla podmínkou. Proto třída *DataSet* byla nahrazena třídou *XmlDocument*. Tato třída je vytvořena specielně pro práci s XML a lze v ní jednoduše vytvářet hierarchickou strukturu.

5.4.3 Ukládání citátů ve formátu RTF

Databáze na počátku ukládala citáty ve formátu RTF. Tento přístup měl tu výhodu, že citát byl uložen přesně ve formátu, v kterém byl napsán. U každého citátu šel nastavit typ písma, jeho velikost, barva, šlo i nastavit, kde je písmo tučné, kde podtržené atd. Na druhou stranu ukládání citátů ve formátu RTF mělo i velké nevýhody. Samotný citát v databázi zabíral příliš velké místo. Soubor databáze nebyl příliš čitelný. Navíc se ukládala spousta informací, které nebylo třeba ukládat, jako např. RTF hlavička. Proto bylo od ukládání ve formátu RTF upuštěno. Byl vytvořen vnitřní značkovací jazyk databáze, ve kterém se nyní citáty ukládají. Databáze tak nezabírá tolik místa a je značně čitelnější.

5.4.4 Ukládání databáze slov pro kontrolu pravopisu

Databáze slov pro kontrolu pravopisu měla původně také jinou strukturu. Pro každé slovo byl vytvořen element, uvnitř kterého se toto slovo nacházelo. Při načtení takto navržené databáze se pak mezi slovy dané délky a stejného počátečního písmena pohybovalo jako v poli. Velikost databáze ale byla příliš velká. Byl tedy vytvořen nový přístup. Všechna slova se nyní ukládají do jednoho elementu a mezi sebou jsou oddělena zpětným lomítkem. Databáze tak má poloviční velikost.

6 Reakce uživatelů a možná vylepšení

6.1 Reakce uživatelů

Zhodnocení aplikace bylo ponecháno uživatelům, neboť o ně jde především. Uživatelé zhodnotili aplikaci použitelnou. Nejvíce se jim líbila možnost nastavení úrovně pohledu při zobrazení databáze a možnost vytvoření si vlastního rozložení oken. Kladně byla hodnocena i možnost zvětšování písma, kontrola pravopisu či uzamykání oken. Dále se jim líbila možnost ukládat dodatečné informace k jednotlivým citátům.

Uživatelé usoudili, že by bylo vhodné, kdyby se u předem nastavitelných rozložení dal měnit poměr jednotlivých oken. Také by se jim líbilo, kdyby se zobrazovala nápověda u jednotlivých popisků položek v okně pro vkládání.

6.2 Možná vylepšení

Aplikace poskytuje stále ještě spoustu vylepšení. Jedním z nich může být urychlení vyhledávání. To by mohlo být realizováno pomocí indexace obsahu jednotlivých citátů. Došlo by tak k rychlejšímu nalezení citátů obsahující hledané slovo. Dále existuje i mnoho možností, jak vylepšit okenní systém aplikace, a to například může být vytvoření interakce mezi jednotlivými okny, okna se můžou k sobě přichytávat atd. Také se může vytvořit možnost změnit poměr mezi okny u přednastavených rozložení. Mezi možná vylepšení může patřit i vytvoření nových možností pro uživatele. Například okno s citátem by se mohlo klonovat, tím by uživatel dostal možnost si některé citáty nechat zobrazeny na pracovní ploše aplikace. Mohl by si tak předpřipravit několik citátů pro pozdější použití.

7 Závěr

Tato práce mi byla velkým přínosem. Díky ní jsem si mohl vyzkoušet, co všechno obnáší tvorba uživatelského rozhraní. Také se rozšířily mé obzory v oblasti současných trendů této tvorby.

Mým cílem bylo vytvořit hlavně použitelnou aplikaci. To jak moc je aplikace použitelná se ukáže časem, až ji někdo bude delší dobu využívat ke sběru citátů. Nicméně první ohlasy od uživatelů byly převážně pozitivní. Proto si myslím, že aplikace může být v praxi použitelná.

Při implementaci bylo dosaženo splnění základních požadavků na GUI. Aplikace poskytuje i několik možností, které by mohly práci uživateli zpříjemnit. Povedl se mi také implementovat jednoduchý okenní systém. Jsem tedy s výsledkem spokojen.

Aplikaci lze dále vyvíjet a vylepšovat. Je možné implementovat nové funkce, které by uživatel mohl ještě využít či vylepšit okenní systém. Některá vylepšení jsou popsána v kapitole 5.2 – Možná vylepšení.

Literatura

[1] Reimer, Jeremy. *A History of the GUI* [online]. 2005-05-05. Dostupné z: <http://arstechnica.com/articles/paedia/gui.ars>

[2] Petřek, Jiří. *Výběr redakčního systému - GUI (2. část)* [online]. 2006-08-27. Dostupné z: <http://blog.jur4.net/44-vyber-redakcniho-systemu-gui-2-cast.html>

[3] Prokop, Marek. *Trendy moderního webdesignu - LUPA* [online]. 2003-08-08. Dostupné z: <http://www.lupa.cz/clanky/trendy-moderniho-webdesignu>

Seznam použitých zkratek

angl. - anglický

GUI - Graphical User Interface

HTML - Hypertext Markup Language

PARC - Palo Alto Research Center

RTF - Richt Text Format

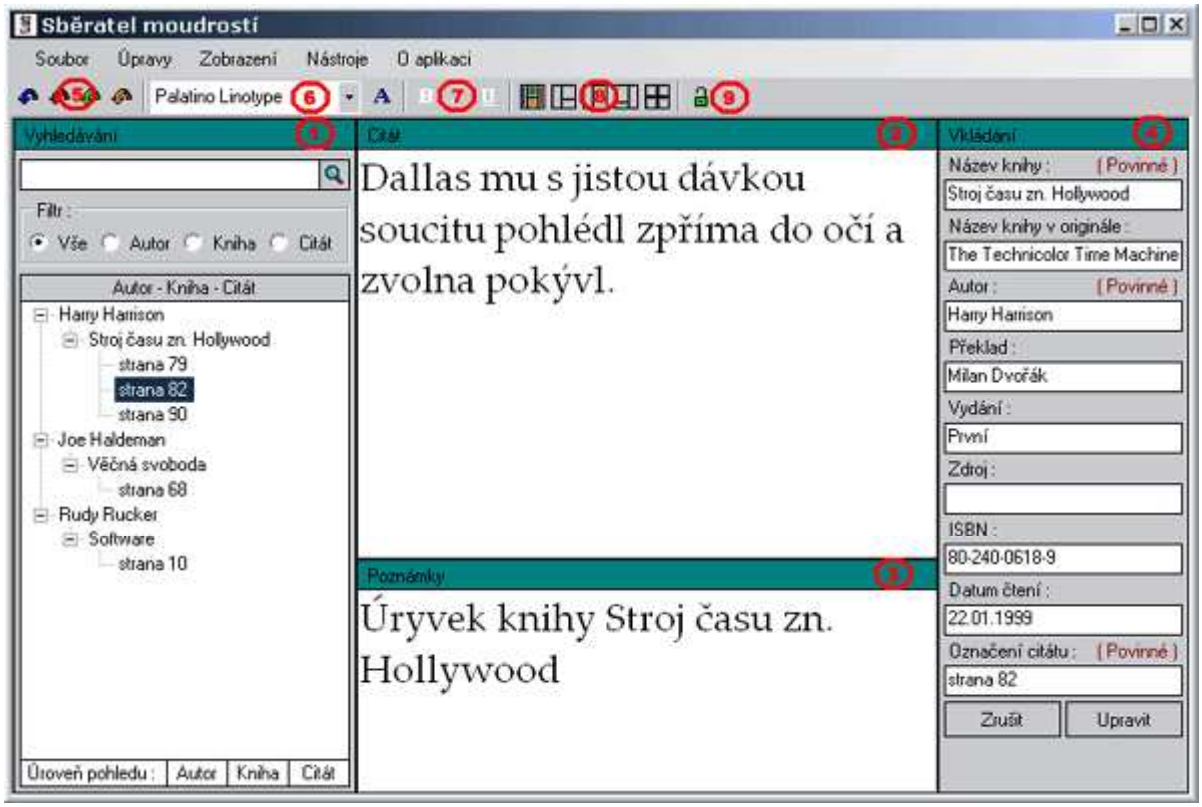
WYSIWYG - What You See Is What You Get

XML - Extended Markup Language

Seznam příloh

Příloha 1. Manuál

Popis aplikace:



1. **Okno pro vyhledávání** – v tomto okně vidíme databázi citátů. Na databázi je možné použít filtr a nechat si tak zobrazit jen část databáze.
2. **Okno s citátem** – v tomto okně se zobrazuje vybraný citát.
3. **Okno s poznámkami** – v tomto okně se zobrazují poznámky k citátu.
4. **Okno pro vkládání** – v tomto okně se vyplňují údaje o knize
5. **Panel zapínání/vypínání** – v pomocí tohoto panelu je možné jednotlivá okna zapínat/vypínat
6. **Panel pro nastavení prostředí** – zde si můžeme zvolit, jakým typem písma a jakou barvou se mají citáty zobrazovat
7. **Panel editace textu** – při ukládání citátů si můžeme určitý text odlišit pomocí kurzívy, tučného nebo podtrženého písma
8. **Panel pro nastavení rozložení oken** – využitím tohoto panelu lze přepínat mezi čtyřmi přednastavenými rozloženími oken a svým vlastním rozložením
9. **Tlačítko na uzamknutí/odemknutí oken** – pomocí tohoto tlačítka je možné okna zafixovat či povolit práci s nimi

Práce s databází

Popis okna pro vyhledávání:

1. **Textové pole** pro vložení filtrujícího textu
2. **Tlačítko na použití filtru** – pomocí tohoto tlačítka se použije filtr na databázi. V databázi se zobrazí jen položky, které filtrující text obsahují.

   Pomocí tohoto tlačítka lze také nastavit, zda se má filtr použít po zmáčknutí tlačítka nebo inkrementálně při psaní filtrovaného textu. Děje se tak pomocí kontextového menu tohoto tlačítka.
3. **Filtr** – v této části se nastavuje, kde se má filtrovaný text nacházet
4. **Typ zobrazení databáze** – tímto tlačítkem si přepínáme mezi dvěma typy zobrazení databáze. První je typ Autor-Kniha-Citát, druhý typem je Kniha(Autor)-Citát.
5. **Kontextové menu položek databáze** – každá položka databáze má kontextové menu, pomocí kterého lze položky editovat nebo mazat. Kontextové menu položek na úrovni citátů obsahuje pouze možnost mazání. Editace citátů se provádí pomocí okna pro vkládání.
6. **Přepínač úrovně pohledu** – pomocí těchto tlačítek lze stromovou strukturu databáze rozbalit do určité úrovně

Přidávání a editace citátů

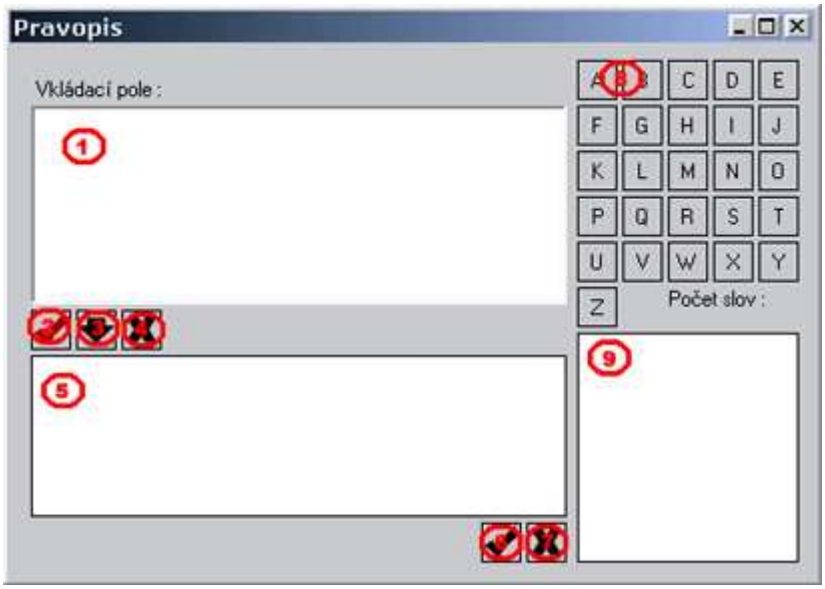
Obsah citátu se vpisuje do **okna citátu**, poznámky pak do **okna poznámek**. Údaje o knize se vyplňují v **okně pro vkládání**. Aby přidání citátu proběhlo úspěšně, musí být vyplněny povinné položky. Při přidávání citátů si můžeme část textu zvýraznit pomocí **panelu editace textu**.

Po přidání citátu do databáze se vyčistí **okno s citátem** a **okno s poznámkami**. V **okně pro vkládání** je smazána jen položka *Označení citátu*. Můžeme tedy přidávat více citátů téže knihy zároveň a nemusíme pokaždé vyplňovat údaje o knize. Tlačítkem *Zrušit* inicializujeme všechny položky.

Chceme-li citát editovat, musíme mít zapnuto rozložení s **oknem pro vkládání**. Pak si stačí citát jen zobrazit a provést požadované změny. Editaci potvrdíme tlačítkem *Upravit*.

Kontrola pravopisu

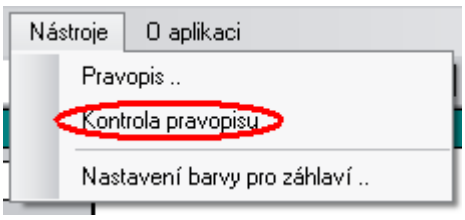
Popis okna pro vkládání nových slov pro kontrolu pravopisu:



1. **Vkládací pole** – do tohoto pole se vpisují nová slova, která mají být přidána do databáze. Jednotlivá slova se oddělují členícími znaménky. Toto pole se také používá pro vkládání textu, který chceme zanalyzovat
2. **Analýza** – tímto tlačítkem analyzujeme text uvnitř vkládacího pole. Slova neznámá pro databázi zčervenají a přidají se **seznamu potencionálních slov**
3. **Vytvořit seznam potencionálních slov** – text vložený uvnitř vkládacího pole se rozdělí na jednotlivá slova a ty se přidají do **seznamu potencionálních slov**
4. **Smazat** – smaže označená slova ze **seznamu potencionálních slov**
5. **Seznam potencionálních slov** – zde se zobrazují slova, která mají být přidána do databáze
6. **Přida**t – tímto tlačítkem přidáme slova do databáze
7. **Zrušit** – pomocí tohoto tlačítka zrušíme přidávání do databáze
8. Tlačítka umožňující **zobrazení slov daného písmene** v databázi
9. **Slova v databázi**

Kontrola pravopisu

U textu citátu nebo poznámek si můžeme zkontrolovat pravopis. To je možné pomocí položky v menu *Nástroje-Kontrola* pravopisu nebo klávesovou zkratkou **ctrl + K**. Slova, která kontrolou neprošla, se zbarví dočervena.



Umístění databáze slov pro kontrolu pravopisu

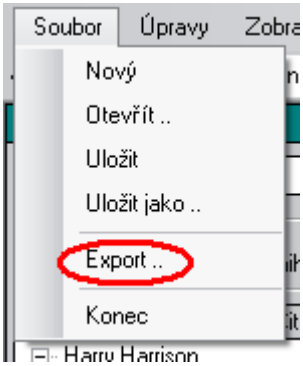
Databáze slov pro kontrolu pravopisu je umístěna ve složce „*OrthDat*“, která se nachází ve složce s aplikací.

Zvětšování písma citátu

Zvětšování písma se provádí v **okně s citátem** nebo v **okně s poznámkami**. Pro zvětšení písma stiskneme pravé tlačítko myši a pohybujeme kurzorem směrem vzhůru. Chceme-li písmo zmenšit, pohyb kurzoru směřuje směrem dolů.

Export citátů do HTML

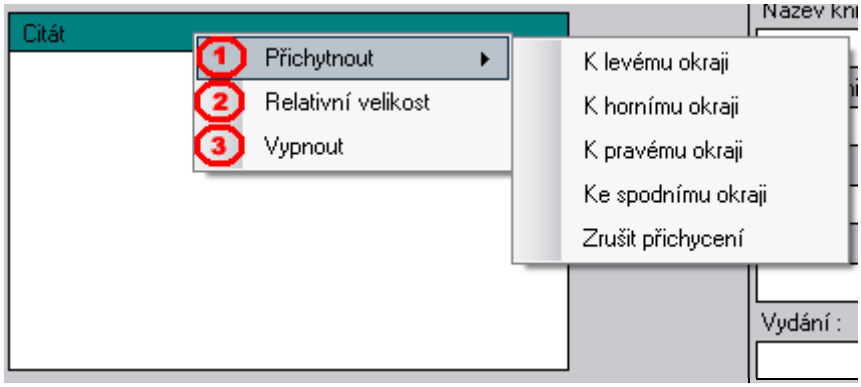
Zobrazený citát můžeme exportovat do HTML. Děje se tak pomocí položky *Soubor – Export* nebo pomocí klávesové zkratky **ctrl + E**. Exportovat do HTML lze pouze zobrazený citát.



Vytvoření vlastního rozložení aplikace

K vytvoření vlastního rozložení využijeme okenního systému. Pohyb a změna velikosti jednotlivých oken se provádí obdobně jako ve Windows. Vlastnosti oken lze upravit pomocí jejich kontextových menu.

Popis kontextového menu oken:



1. **Přichytnout** – pomocí této položky lze okno přichytnout k okraji aplikace
2. **Relativní velikost/Absolutní velikost** – pomocí této položky lze nastavit, aby se velikost okna měnila s velikostí aplikace nebo zůstala konstantní
3. **Vypnout** – touto položkou se okna vypínají

Klávesové zkratky

1. **ctrl + X** – export citátu do HTML
2. **ctrl + F** – vyhledá se další výskyt označeného textu uvnitř citátu/poznámek
3. **ctrl + shift + F** – hledání
4. **ctrl + K** – kontrola pravopisu
5. **ctrl + O** – načtení databáze
6. **ctrl + N** – nahrazení
7. **ctrl + S** – uložení databáze
8. **ctrl + Y** – opakovat psaní
9. **ctrl + Z** – zpět při psaní