VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

NÁSTROJ PRO UŽIVATELEM URČENÝ PUBLIKAČNÍ VÝSTUP ZE VSTUPNÍCH TEXTOVÝCH DAT

TOOL FOR USER-DEFINED PUBLICATION OUTPUT FROM INPUT TEXT DATA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MICHAL NOVÁK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JOSEF STRNADEL, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

155107

Ústav: Ústav počítačových systémů (UPSY)
Student: **Novák Michal**
Program: Informační technologie
Název: **Nástroj pro uživatelem určený publikační výstup ze vstupních textových dat**
Kategorie: Uživatelská rozhraní
Akademický rok: 2023/24

Zadání:

1. Proveďte rešerši v oblasti informačních zdrojů. Zaměřte se zejména na zdroje primárně určené k tisku (články, knihy aj.) resp. k interakci (webové stránky, prezentace aj.), charakterizujte je a shrňte typické případy jejich užití.
2. Proveďte rešerši v oblastech reprezentace dat a formátů jejich uložení; zaměřte se zejména na textovou podobu dat (TXT, CSV, JSON aj.) a na podobu dat (JPG, PDF, (La)TEX aj.) vhodnou k publikaci výsledků (ilustrace, tabulky aj.) v různých informačních zdrojích.
3. Navrhněte architekturu a mechanismus činnosti softwarového nástroje splňujícího následující požadavky: i) uživatelská přívětivost, modulárnost a rozšiřitelnost, ii) schopnost načíst libovolná textová data splňující uživatelem danou specifikaci, iii) možnost uživatele určit, která z načtených dat chce publikovat, zda a jak je chce anotovat, jaký vzhled (tabulka, obrázek, samostatně, s homo/heterogenní, lineární/maticovou strukturou atp.), vlastnosti (rozměr(y), poměr stran aj.) a formát výstupních dat (PDF aj.) požaduje a v jakém typu informačního zdroje plánuje publikovat, iv) schopnost generovat člověku srozumitelný výstup z bezesporného/platného vstupu. Shrňte případy užití navrženého nástroje.
4. Na základě rozboru zvolte prostředky a metody vhodné k tvorbě a činnosti navrženého nástroje a nástroj s jejich pomocí implementujte. Proveďte základní testování nástroje a připravte podmínky pro ověření jeho vlastností a jeho zhodnocení.
5. Vlastnosti implementovaného nástroje vhodně ověřte, zejména co se týká splnění požadavků z bodu 3 v různých případech užití pro publikování v různých typech informačních zdrojů.
6. Zhodnoťte vlastnosti implementovaného nástroje a výsledky pomocí něj dosažené; identifikujte silné a slabé stránky implementace, navrhněte možné směry jejího vylepšení a rozveďte ty, které považujete za nejvíce perspektivní.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání,
- představení kandidátních prostředků a metod pro realizaci navrženého systému.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Strnadel Josef, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 30.10.2023

Abstrakt

Tato bakalářská práce se zaměřuje na návrh a implementaci nástroje určeného k tvorbě publikačních výstupů ze vstupních textových dat. Cílem práce je navrhnout a vytvořit univerzální a modulární program, který za pomoci uživatelské konfigurace dokáže převést surová (serializovaná) data do vizuální reprezentace, jež může být nadále použita v publikacích. V práci jsou zkoumány informační zdroje, jejich dělení a použití. Dále je proveden rozbor běžných formátů pro serializovaná a vizualizovaná data, doplněný o metody a přístupy k vizualizaci dat. Jsou zde vysvětleny jednotlivé části a přístupy k návrhu nástroje, na které navazuje popis implementace. Na závěr je provedeno uživatelské testování a zhodnocení kvality implementace.

Abstract

This bachelor's thesis focuses on the design and implementation of a tool intended for creating publication outputs from input text data. The aim of this thesis is to design and implement a universal and modular program that can transform raw (serialized) data supplemented with user configuration into visual representation that can then be used in publications. Thesis explores sources of information, their types, and usage. Additionally, common formats for serialized and visualized data are analysed and a research of methods and approaches to data visualization is conducted. The individual components and approaches to tool design are explained, followed by a description of implementation. At the end, the tool is tested by users. Results of these tests are then used for evaluation of the quality of implementation.

Klíčová slova

informační zdroj, tištěný pramen, elektronický pramen, formát dat, formát, serializace dat, vizualizace dat, diagram, graf, software, datová analýza

Keywords

information source, print source, electronic source, data format, format, data serialization, data vizualization, diagram, graph, software, data analysis

Citace

NOVÁK, Michal. *Nástroj pro uživatelem určený publikační výstup ze vstupních textových dat.* Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Josef Strnadel, Ph.D.

Nástroj pro uživatelem určený publikační výstup ze vstupních textových dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Josefa Strnadela, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Novák
25. dubna 2024

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Josefu Strnadelovi, Ph.D. za jeho pomoc, odborné rady, nápady, drahocenný čas a hlavně trpělivost. Dále bych rád poděkoval všem, kteří se podíleli na testování výsledné implementace, za jejich pomoc a přátelský přístup.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Vědecké práce a výzkumy s sebou mnohdy přináší i řady pokusů, jejichž výsledky musí být nějak zaznamenány. Záznamy jsou většinou uloženy v podobě některého z rozšířených, nebo i atypických serializačních formátů. Ovšem taková data nemusí pocházet jen z výzkumů a vědeckých prací. Data mohou být generována například dotazníkovými šetřeními, přístroji pro snímání okolí, programy pro monitorování sítě a mnohými dalšími způsoby.

Lidský mozek však jen obtížně dokáže zpracovat velké množství dat reprezentovaných v surové podobě. Proto tuto práci necháváme na počítačích a vhodném softwaru, jenž dokáže data načíst, zpracovat a vytvořit „jednoduchý“ vizuální výstup srozumitelný pro lidský mozek. Proč se ale vůbec snažíme vytvořit vizuální výstup srozumitelný lidem? Prvním z důvodů je pochopení dat pro vlastní potřebu. Snažíme se data udělat pochopitelná pro nás samotné. Ovšem častěji to možná děláme z toho důvodu, že svoje výsledky chceme někomu prezentovat, což je obvykle prováděno pomocí publikací.

Nástrojů pro zpracování a vizualizaci dat rozhodně není málo a každým dnem se jejich počty zvyšují. Ovšem tyto nástroje mají většinou pevně definované formáty vstupních dat či grafickou podobu a formát vizuálního výstupu. Zároveň s tím existují lidé, kteří by si právě tyto problémy mohli sami vyřešit například vlastními rozšířeními původního nástroje.

Cílem práce je tedy vytvořit jednoduchý, ale zároveň univerzální nástroj, jehož funkcionalita může být jednoduše rozšířena vytvořením vlastních modulů. Nástroj by měl dále splňovat všechny základní funkcionality rozšířených vizualizačních nástrojů, tj. načtení dat, jejich zpracování a vizualizace. Práce poskytne přehled o informačních zdrojích včetně jejich běžných uplatnění a podob. Dále se práce bude věnovat problematice formátů serializace dat, na niž plynule naváže téma vizualizace dat a běžných přístupů k vizualizaci.

Naplnění požadavků na implementaci je na konci práce ověřeno pomocí uživatelského testování. Výsledky testů jsou vyhodnoceny a doplněny o vlastní zhodnocení vlastností implementace spolu s možnými návrhy na další práci.

Důvodem pro výběr tématu byl zájem navrhnout a implementovat softwarový nástroj, který splňuje specifické požadavky zadání. Vzhledem k tomu, že zadání není přesně definované, umožňuje zapojit trochu kreativity a vlastního pochopení problému. Je to skvělá příležitost k rozvoji v oblasti softwarového inženýrství.

Kapitola 2

Informační zdroje

Informační zdroj je prostředkem komunikace a jako takový je tvořen množinou informací. Slouží pro záznam a přenos informací v čase a prostoru.

Informační zdroje je možné dělit více způsoby. Jedním z těchto způsobů je dělení dle typu nosiče. Rozlišují se zde **tištěné**, **elektronické** a **mikrografické** prameny. Další možné rozřazení je podle původnosti obsahu. Vlastní informace jsou přinášeny **primárními informačními prameny**. O existenci primárních informačních pramenů informují **prameny sekundární**. Nejedná se o celé texty, ale pouze jen odkazy. **Terciální informační prameny** dále informují o sekundárních pramenech. [28] Tato kapitola se bude dále zabývat převážně primárními informačními prameny, a to v tištěné a elektronické podobě.

2.1 Tištěné prameny

2.1.1 Historie knih a uchování informací

Již od vzniku písma sloužila kniha jako způsob uchování a přenosu informací či zpráv všeho druhu. Původ slova „kniha“ je sám o sobě nejasný. Podle jedné z teorií se může jednat o derivát slova asyrského původu „kunukku“, které v překladu znamená „pečeť“. [50] Samotná podoba knihy se měnila napříč časem a prostorem. Za její nejstarší podobu lze považovat hliněné destičky popsané klínovým písmem.

S vývojem civilizace přicházelo lidstvo na lepší způsoby zápisu informací. Velkou změnu přinesl vynález papyru, který se vyráběl z pásků rákosu. Oproti hliněným destičkám byl velmi lehký a skladný. Papyrus je však křehký a není možné jej překládat. Z toho důvodu byl motán do svitků.

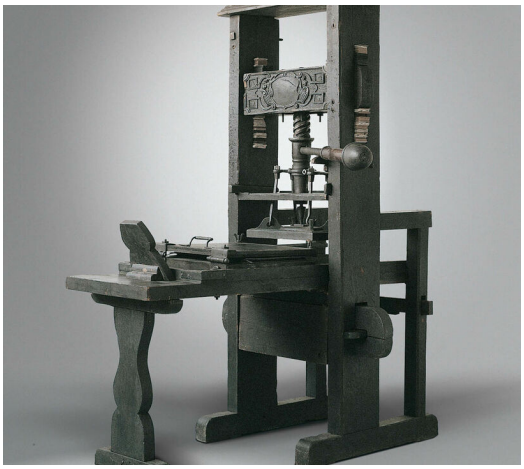
Dalším výrazným posunem byl vynález pergamenu. Pergamen byl vyráběn ze sušené a očištěné kůže. Oproti papyru měl několik výhod. Byl ohebný, odolný a dalo se na něj psát z obou stran. V případě potřeby se dal „seškrábat“ a znovu popsat.

Největším pokrokem však byl vynález papíru, materiálu, který je používán dodnes. Papír se začal šířit z Číny od počátku 2. století n.l. a byl vyráběn z hedvábných a lněných hadrů. V Evropě se papír vyskytl mnohem později, a to kolem 12. století, kdy nahradil do té doby používaný pergamen.

Obvyklým způsobem šíření knih bylo přepisování. Tento velmi náročný proces výrazně zvyšoval jejich hodnotu. Vlastnictví knih se tedy spojovalo s velkým majetkem a vysokým postavením ve společnosti. [31] To se ovšem změnilo v polovině 15. století představením knihtisku, za jehož vynálezce je považován Johannes Gutenberg. V této souvislosti je ovšem nutno poznamenat, že v Číně bylo již od konce 6. století využíváno tisku za pomoci dřevěné

matrice. V 11. století se ve stejné oblasti dále přešlo na tisk jednotlivých znaků pomocí razítek. O těchto metodách se ovšem v Evropě příliš nevědělo.

Podstatou převratného Gutenbergova vynálezu bylo uspořádání jednotlivých písmen do tiskové formy, přičemž po ukončení tisku byla písmena rozebrána a připravena opět k použití do nové tiskové formy. Pokrokovou myšlenkou bylo rovněž použití kovu pro tvorbu odlitků písmen (tzv. liter). [51]



Obrázek 2.1: Dřevěný knihtiskový lis. Převzato z [5]

Obrázek 2.2: Kovové tiskařské litery. Převzato z [30]

Rozvoj knihtisku brzy umožnil vznik prvních tiskáren, čímž došlo k velkému snížení cen knih. Ty se tak začaly šířit i mezi nižší vrstvy obyvatel, což vedlo ke zvýšení gramotnosti obyvatel. Stále a stále docházelo k vylepšování tisku až do podoby, která je známa dnes.

S nástupem počítačů ke konci 20. století přichází období digitalizace. Nové knihy se již vytvářejí v elektronické podobě a až poté bývají převedeny do tištěné formy. Informace jsou dnes dostupnější než kdy dříve, doslova „za pár kliknutí na internetu“.

2.1.2 Kniha jako písemný dokument

Knihu dnes můžeme definovat jako duševní dílo tvořící myšlenkový a výtvarný celek s minimálním rozsahem 48 stran, který nevychází periodicky. Dále musí mít přiděleno ISBN. Mezi knihy se řadí monografie, učebnice, skripta, příručky a sborníky. [28]

Monografie neboli odborná kniha slouží k prezentaci původních výsledků výzkumu. Autorem či autory jsou poté zpravidla lidé, kteří se na něm podíleli. V textu lze očekávat odkazy a seznam použité literatury. Povinností je i souhrn v alespoň jednom ze světových jazyků. [35]

Sborník je možno podle jedné z možných definic popsat následovně: „publikace vzniklá spojením autorsky různorodých, žánrově i obsahově však příbuzných a rozsahem zpravidla nevelkých literárních děl.“ [49] U každého z dílčích děl zpravidla bývá uvedeno jméno autora, které se může vyskytnout i na společné obálce.

Skripta a učebnice jsou obecně učební texty určené ke vzdělávání. Učebnice na rozdíl od skript, která se používají převážně na vysokých školách, musí prvně projít schvalovacími procesy.

Příručka je termín, pod který se řadí encyklopedie, slovníky, klíče, průvodce, adresáře, návody a další. [28] Příručku je možné definovat jako „dokument umožňující rychlý

a efektivní přístup k informacím, znalostem a dalším primárním zdrojům na základě systemizované prezentace základních znalostí z určitého vědního oboru nebo z určité oblasti lidské činnosti.“ [39]

2.1.3 Periodika

Periodikum lze charakterizovat jako dokument, který je vydáván postupně po částech. Jednotlivé části zpravidla nesou shodný název a bývají doplněny o pořadové číslo či datum vydání. Intervaly mezi vydáním mohou být pravidelné i nepravidelné. [38] Jako periodika se označují noviny, časopisy, periodické sborníky a ročenky.

Noviny vycházejí s periodicitou jednoho dne až týdne a jejich účelem je informování široké veřejnosti o aktuálních událostech. Časopisy už nevycházejí tak často jako noviny, ale oproti tomu mají lepší úpravu a bývají tematicky zaměřené. Periodické sborníky jsou často publikovány vysokými školami nebo odbornými institucemi pod společným názvem v pravidelných či nepravidelných intervalech a obsahují vědecké studie pracovníků dané instituce. [37] Ročenka „poskytuje přehledné informace různého druhu, zejm. představuje výsledky činnosti za příslušný rok, a to ve stručné popisné nebo statistické formě a ve vztahu k určité zemi, regionu, instituci, oboru nebo předmětu (např. ročenky statistické, událostí roku, bibliografické, fotografické, institucionální aj.).“ [40]

2.1.4 Speciální dokumenty

Speciální dokumenty jsou rozsáhlou kategorií textů, které nevycházejí běžnou formou. Patří zde například vědecko-kvalifikační práce, materiály z konferencí, výzkumné zprávy.

Vědecko-kvalifikační práce zahrnují práce, které slouží jako podklad pro udělení akademické, vědecké nebo pedagogické hodnosti. Materiály z konferencí obsahují příspěvky přednesené na konferenci. Výzkumné zprávy jsou používány k prezentaci výsledků řešení výzkumů. [28]

2.2 Elektronické prameny

Od vynálezu papíru uběhla již dlouhá doba a i když je stále používán, není již dominantním způsobem uchování informací. Technologický pokrok přinesl v posledním století nespočet jiných, a to elektronických nosičů informací.

Elektronický informační zdroj je takový zdroj, který je uchováván v elektronické podobě a lze k němu přistoupit za pomoci počítačových sítí nebo jiných technologií distribuce digitálních dat. [27] Sama o sobě je tato definice vcelku pochopitelná, ale pro upřesnění je možné použít i vysvětlení: „Základním a určujícím rysem elektronických informačních zdrojů je skutečnost, že obsahují informace zaznamenané na elektronickém nosiči (paměť počítače, disketa, CD-ROM, apod.) a nejsou pro člověka bezprostředně čitelné. Jejich vnímání a užití je možné pouze pomocí technického zařízení, tj. počítače, který je převede do podoby textové, obrazové, zvukové nebo zvukově obrazové.“ [44]

Elektronické informační zdroje je možno dělit podle přístupnosti na offline zdroje a online zdroje. Online zdroje jsou dostupné za použití počítačových sítí v rámci internetu a přístup k nim je téměř okamžitý. I přes to, že je velká část z nich dostupná volně a bez poplatku, existují i zdroje, které jsou skryté před neoprávněnými osobami nebo je přístup k nim zpoplatněn. Z online zdrojů se mohou stát offline zdroje stažením do lokálního počítače nebo na jiný nosič. Offline zdroje jsou tedy naopak dostupné bez nutnosti připojovat se

do počítačových sítí. Mimo stažení je možno je distribuovat přímo s nosičem (např. hudba na CD).

Elektronické zdroje jsou pro mnohé příznivější variantou zisku informací, ať už z důvodu manipulace, vyhledávání či ceny za poskytnutí a uskladnění. V některých případech neexistuje ani jiná než elektronická varianta. Ceny tištěných publikací vedou některé instituce k přechodu do digitální podoby. [19]

2.2.1 Internet jako zdroj informací

Využívání internetu kompletně mění způsob, jakým lidé žijí a učí se.

Stručná historie

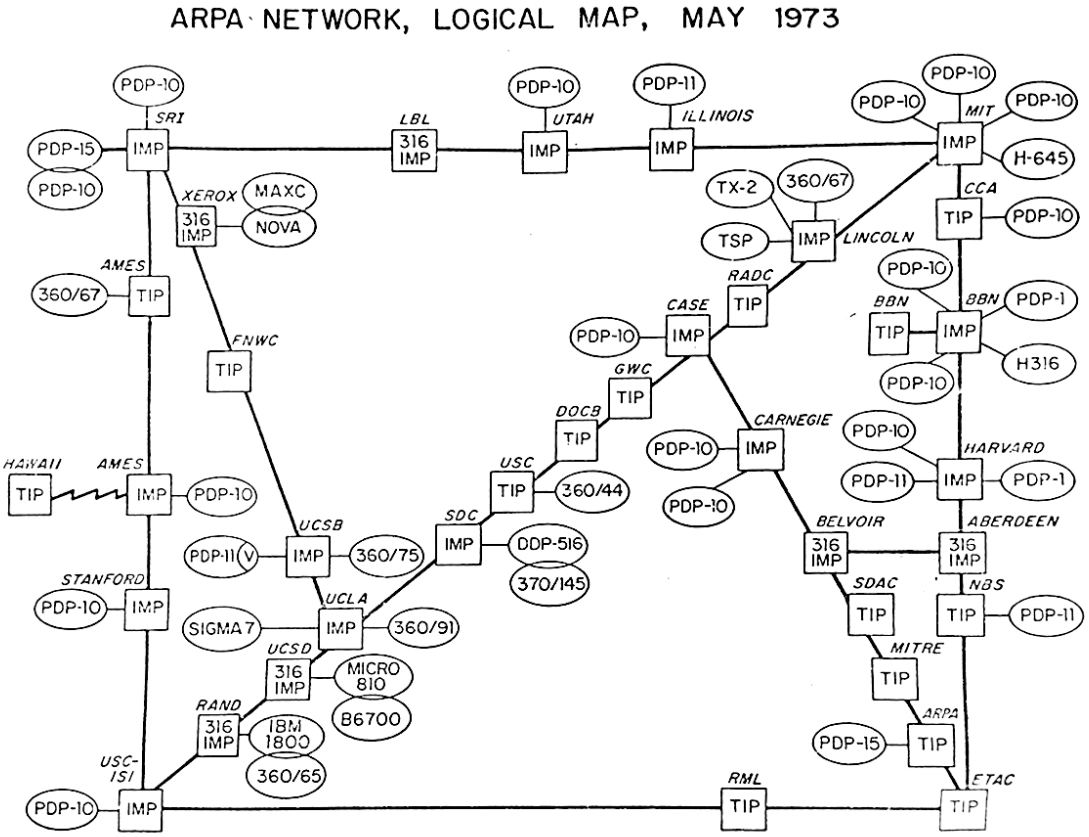
Vše začalo na popud amerického prezidenta Eisenhowera vytvořením Úřadu pro projekty pokročilého výzkumu (the Advanced Research Projects Agency, ARPA), který vznikl roku 1958 jako reakce na vypuštění družice Sputnik během studené války. Nastaly obavy o tehdejší způsob komunikace, jenž by mohl být v případě jaderné války jednoduše odstaven jadernými útoky. S řešením tohoto problému měl přijít právě úřad ARPA.

Počítače tehdy byly výrobci stále chápány jako aritmetické stroje. Výzkumníci úřadu ARPA však počítač chápali jako komunikační médium mezi lidmi. Bylo tedy nutné se spojit s počítačovým průmyslem.

V rámci výzkumu vznikly počáteční protokoly, jazyky či procesory pro počítačovou komunikaci. Pořádala se zasedání, na kterých byly jednotlivými skupinami diskutovány možné přístupy. Tato zasedání byla ovšem obtížná, jelikož každá skupina měla jiné názory a připomínky. Vznikly tedy dokumenty RFC (Request for Comments). Výzkum se nakonec vyplatil a na podzim roku 1969 vznikla síť ARPANET. [29]

Síť ARPANET už nebyla tak lehce zranitelná jako předchozí metoda komunikace, která veškerou komunikaci vedla přes 1 uzel. Místo toho vzniklo několik uzlů, které byly navzájem rovnocenné. I přes tuto skutečnost síť stále nesplňovala požadavky ministerstva obrany USA, jelikož byl přenos dat přes uzly nespolehlivý a často vypadával. Data se přenášela vcelku, což způsobovalo problémy. Vznikla myšlenka tato data rozdělit na části nazývané pakety. Každý paket měl v sobě obsahovat adresu příjemce. Cesta se následně určila až podle aktuálního vytížení a provozuschopnosti sítě. Pakety se předávaly mezi jednotlivými funkčními uzly, dokud nedorazily k adresátovi.

Spolehlivost a funkčnost došla do takového stavu, že vyhovovala požadavkům armády Spojených států a přešlo se do stavu realizace. Síť měla původně omezený přístup jen pro armádní účely, ale vzhledem k rychlému rozvoji a rozšíření bylo omezení zrušeno. K síti se postupně připojovaly různé univerzity a velké organizace, až nakonec došlo i k rozšíření mimo území Spojených států do Velké Británie. Síť se neustále rozvíjela a rostla až do podoby dnešního internetu. [34]



Obrázek 2.3: Schéma sítě ARPANET, květen 1973. Převzato z [17]

2.2.2 Internetové stránky

Internetové stránky, též nazývané jako webové stránky nebo www stránky, jsou dokumenty dostupné pomocí internetového prohlížeče. Stránky se svým návštěvníkům prezentují estetickým a čitelným způsobem. Webové stránky se běžně používají k propagačním účelům, ale slouží i jako významný zdroj informací.

Přístupy na webové stránky je možno omezit několika způsoby. Pokud není důvod stránky zpřístupňovat přes celý internet, mohou být dostupné jen v rámci lokální sítě. Další možností je povolení přístupu registrovaným uživatelům, kteří přístup získali od provozovatele stránek (např. na základě platby).

V dnešní době může vytvářet webové stránky téměř každý. Jsou tvořeny soustavou textů, obrázků a multimediálního obsahu. Jednotlivé stránky na sebe mohou odkazovat za pomoci unikátních adres.

2.2.3 Důvěryhodnost

Ze začátku je důležité zmínit, že tento problém není relevantní čistě jen pro elektronické zdroje. I v případě tištěných pramenů je potřeba mít kritické smýšlení a hledat důvěryhodné zdroje informací.

Setkat se s chybnými nebo nepřesnými informacemi na internetu není v dnešní době nic složitého. Přeci jen, informace v tomto prostředí může šířit téměř každý. Internet se navíc rozrostl do velikosti, ve které není přijatelné a ani možné uplatnit větší plošné regu-

lace. Ale i přesto, že se internet stal informačním zdrojem, který obsahuje velké množství nepravdivých údajů, je možno z něj čerpat i důvěryhodné informace. Jen je potřeba vědět jak.

Profesionálně vypadající stránka ještě není zárukou kvality zdroje. Existuje velké množství stránek, které napodobují jiné známé stránky, například za účelem zisku přihlašovacích údajů. Vizuálně se přitom stránky vůbec neliší, jediným rozdílem je jen adresa stránek. Pokud adresa stránky souhlasí, stále není možno se plně spoléhat na obsah. Zdroje mohou být nespočet způsoby napadeny a vandalizovány. [48]

Neexistuje tedy přesný návod, jak rozpoznat důvěryhodný zdroj. Obvykle záleží čistě na intuici a zkušenostech. Obecně platí pravidlo ověřování informací na více nezávislých stránkách. Dále je vhodné čerpat informace z míst, která odkazují na důvěryhodné dokumenty.

Kapitola 3

Reprezentace dat a formát jejich uložení

Formát souboru

Formát souboru je pojem, který se vztahuje k technickému standardu, jenž je používán k ukládání informací v počítačovém souboru. Formáty mohou být navrženy pro jeden či více specifických typů dat a nejčastěji lze formát rozpoznat dle přípony souboru. Dále se informace o formátu mohou nacházet ve vnitřních a vnějších metadatech, což jsou data poskytující informace o primárních datech. [6]

Většina dnes běžně používaných formátů je popsána vlatním standardem. Tímto přístupem se zaručuje kompatibilita, přenositelnost a znovupoužitelnost formátu. Standardizaci mohou provádět různé organizace, mezi něž můžeme zařadit organizace jako IETF nebo ISO.

Na základě standardu mohou vznikat jednotné nástroje, které dokáží provádět čtení a zápis do souboru v požadovaném formátu.

3.1 Serializace dat

V programovacích jazycích se pro uchování dat běžně používají struktury a objekty, které jsou vázány na životní cyklus běhu programu. Pokud je potřeba taková data uložit nebo odeslat, musí se změnit i způsob jejich interpretace, k čemuž slouží právě serializace dat. Obecně se jedná o proces převedení strukturovaných dat do sekvenční podoby. Zvrácení tohoto procesu a převedení dat zpět do vnitřní reprezentace se nazývá deserializace.

Neexistuje však pouze jeden univerzální serializační formát. I když se u jednotlivých formátů mohou vyskytovat společné rysy, bývají často odlišné. Každý formát má vlastní skupinu pravidel a je většinou vhodný pouze pro určitou množinu případů užití. [47]

3.2 Formáty pro uchování serializovaných dat

TXT

Běžný textový soubor je téměř nejuniverzálnějším formátem pro ukládání dat. Informace se zde ukládají v podobě prostého textu. Nemá žádné specifické formátování ani předepsanou strukturu. Editovat jej lze prakticky v každém textovém editoru. Text může být dělen do

pomyslných odstavců pomocí ukončení řádku charaktery CRLF (Carriage Return a Line Feed).

CSV (Comma Separated Values)

CSV je standardizovaný formát, který se běžně užívá k ukládání tabulkových dat. Data jsou uložena v záznamech. Záznamy jsou standardně definovány na jednom řádku končícím charaktery CRLF. Údaje v záznamu jsou od sebe odděleny čárkou. Na prvním řádku může být definována hlavička sloužící k pojmenování jednotlivých sloupců. Pokud je potřeba, aby údaje záznamu obsahovaly čárku, charaktery CRLF nebo dvojité uvozovky, musí být takové záznamy uzavřeny do dvojitých uvozovek. V případě dvojitých uvozovek uvnitř záznamu musí být použito tzv. escape sekvence, kdy se tyto dvojité uvozovky „předřadí“ dalšími dvojitými uvozovkami. [45]

```
sloupec1,sloupec2,sloupec3 CRLF
x1,x2 x3,"x4, x5" CRLF
y1,"yCRLF
2","y""3"
```

Listing 1: Ukázka možného formátování CSV

sloupec1	**sloupec2**	**sloupec3**
x1	x2 x3	x4,x5
y1	y2	y"3

Tabulka 3.1: Interpretace CSV

Ukázku CSV souboru ve výpisu 1 je tedy poté možné interpretovat tak, jak to provádí tabullka 3.1.

JSON (JavaScript Object Notation)

JSON je velmi oblíbeným formátem pro komunikaci přes aplikační rozhraní programovacích jazyků. Oproti jiným serializačním formátům definuje jen malou množinu pravidel. Jeho struktura je odvozena od jazyka JavaScript. V JSONu mohou být reprezentovány základní datové typy string (textový řetězec), čísla, boolean (pravdivostní hodnota) a null, které se mohou sdružovat do struktur typu pole nebo objekt.

Data ve formátu JSON jsou uložena jako dvojice `klíč:hodnota`, přičemž klíč musí být vždy textový řetězec ohraničený dvojitými uvozovkami. Hodnotou dále může být jakýkoliv z výše zmíněných datových typů či struktur. Jednotlivé dvojice musí mezi sebou být odděleny pomocí čárky. Pro deklaraci začátku a konce nového objektu jsou použity složené závorky `{ }`. Seznamy jsou deklarovány hranatými závorkami `[ ]` a jejich prvky jsou opět odděleny čárkou. JSON dále zanedbává bílé znaky mezi prvky struktury. [21]

```
{
  "name": "Michal",
  "languages": ["cz","en"],
  "employed": false,
  "studies": {
    "name": "Brno University of Technology",
    "faculty": "Faculty of Information Technology",
    "year": 3
  }
}
```

Listing 2: Ukázka formátu JSON

XML (Extensible Markup Language)

Jazyk XML patří do rodiny značkovacích jazyků a používá se k serializaci dat. Na rozdíl od běžných značkovacích jazyků, jako je například HTML, neobsahuje XML předem definované značky. Názvy značek tedy mohou být definovány přesně dle uživatelských potřeb, což z jazyka XML dělá skvělý způsob ukládání dat, která mohou být jednoduše prohledávána a sdílena.

Na začátku každého XML dokumentu by se měla vyskytnout hlavička určující verzi XML. V hlavičce se dále může nacházet údaj o použitém kódování a atribut samostatnosti, který uvádí, jestli jsou všechny deklarace entit vyžadované dokumentem XML obsaženy v dokumentu.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<name>Michal</name>
<languages>cz</languages>
<languages>en</languages>
<employed>false</employed>
<studies>
    <name>Brno University of Technology</name>
    <faculty>Faculty of Information Technology</faculty>
    <year>3</year>
</studies>
```

Listing 3: Ukázka formátu XML

3.3 Vizualizace dat

Data se běžně ukládají v surovém nezpracovaném stavu. I když je tento způsob vhodný pro počítačové zpracování, lidský mozek z něj jen těžce získá potřebné informace. Surová data tedy bývají v případě potřeby zpracovávána a převáděna do grafické reprezentace. Tento proces je poté možno nazvat vizualizací dat. V praxi se vizualizace běžně dosahuje pomocí grafů a tabulek. [46]

3.3.1 Přístupy k vizualizaci

Vizualizací se chce běžně dosáhnout nějakého efektu. Aby byl požadovaný efekt pozitivní, musí mít reprezentace přínos pro čtenáře. Z tohoto důvodu se používají ověřené přístupy. [46]

Jedním ze základních předpokladů je znalost publika. Obecenstvo je nutné zaujmout. Musí obdržet informace, které jsou pro něj srozumitelné a přínosné. Není nutné vytvářet složité grafy obsahující každou podrobnost. Je potřeba se přizpůsobit potřebám a očekáváním cílové skupiny. Dalším důležitým předpokladem je jednoduchost. Bývá ve zvyku tvořit graficky komplikované prostorové grafy, ale mnohdy je přínosnější reprezentaci zjednodušit natolik, že je srozumitelná na první pohled. Žádné složité luštění popisků os a legend. Existuje samozřejmě mnoho prostředků na vizualizaci jedné a té samé informace. Ne každá je však efektivní. Je tedy důležité zvolit vhodnou formu reprezentace. [22]

3.4 Formáty uložení vizuálních dat

GIF (Graphics Interchange Format)

GIF je formát, který je dnes převážně známý mezi mladšími generacemi díky sociálním sítím. Jedná se o rastrový formát určený pro ukládání jednoduchých obrázků. Vytvořen byl roku 1987 za účelem urychlení stahování velkých obrázků z internetu. Bylo potřeba vymyslet formát, který je paměťově nenáročný a může být použit na webu, jelikož rychlosti připojení v té době dosahovaly jen malých hodnot. Z toho důvodu je pro barevnou paletu vyčleněno celkem 8 bitů na pixel. Byl tedy stejně paměťově velký jako černobílé obrázky, které také využívaly 8 bitů.

Specifikem pro GIF je možnost tvorby jednoduchých animací pomocí řazení několika snímků za sebe.

JPEG

Za jeden z nejznámějších a velmi rozšířených formátů pro ukládání obrázkových souborů lze považovat právě JPEG. Formát mezi lety 1986 až 1994 vynalezla a standardizovala pod ISO/IEC 10918 mezinárodní organizace nesoucí název Joint Photographic Experts Group, odkud také pochází zkratka označující tento formát. [1]

Standardně se u JPEGu používá 24 bitů pro uložení barvy definované pomocí RGB, ale je možno využít režim 32 bitů pro CMYK nebo 8 bitů pro stupně šedi. Není ovšem podporováno průhledné pozadí. Soubory s ohledem na paletu barev nezabírají příliš úložného prostoru a je možno je v případě potřeby, například odesílání, zmenšit pomocí ztrátové komprese. [8]

V dnešní době je JPEG stále velmi využívaným formátem pro ukládání fotografií a grafiky a zobrazit jej dokáže téměř každý internetový prohlížeč. Soubory mohou mít jednu z přípon `jpg`, `jpeg`, `jpe`, `jif`, `jfif` nebo `jfi`. [10]

PNG (Portable Network Graphics)

PNG je další velmi rozšířený formát, který vznikl jako nástupce formátu GIF roku 1995. Na rozdíl od svého předchůdce (GIF) podporuje 24 bitů uložení barev RGB, na každou barvu je tedy použito 8 bitů. Dalších 8 bitů poté může být použito pro tzv. α kanál, který dodává údaj o transparentnosti. V součtu poté může být 1 pixel definován na 32 bitech. V porovnání

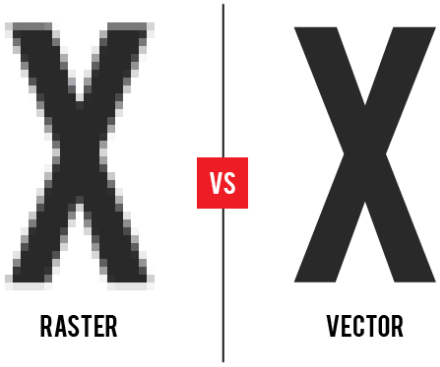
s již zmíněným formátem JPEG podporuje PNG bezztrátovou kompresi a nedochází tedy ke ztrátě dat. [25]

Jedná se opět o oblíbený formát pro tvorbu rastrové grafiky převážně díky možnosti průhledného pozadí. Je opět vhodný i pro webovou grafiku, jelikož je také podporován většinou prohlížečů. [11]

SVG (Scalable Vector Graphics)

Rastrové grafiky mají obecně jeden velký společný problém, a tím je škálovatelnost. Při roztažení rastrového obrázku dochází ke znatelnému rozostření, protože se ukládání informací provádí po pixelech. Oproti tomu vektorové formáty, jak už název napovídá, používají matematické zápisy, kterými jsou definovány jednotlivé objekty grafiky. Při roztažení vektorové grafiky poté jednoduše dochází k přepočítání a vykreslení příslušných pixelů.

SVG je formát vytvořen přímo pro web a vznikl na popud konsorcia World Wide Web Consortium (W3C), které chtělo vytvořit nový typ vektorového grafického formátu. SVG je založeno na jazyce XML (3.2) a je popsáno pomocí množiny 2D vektorů. Pro tvorbu SVG grafiky tedy není potřeba žádný grafický nástroj, nýbrž jednoduchý textový editor na psaní XML a například internetový prohlížeč, který zobrazí SVG v grafické podobě. [13] [15]



Obrázek 3.1: Porovnání rastrové a vektorové grafiky při škálování. Převzato z [26]

EPS (Encapsulated PostScript)

EPS je dalším z vektorových grafických formátů, který vychází z PostScriptu. PostScript (dále jen PS) používá vektorové grafiky pro přípravu textu a obrázků k tisku. Soubory typu PS mohou být poslány přímo k tisku bez nutnosti je otevřít a zpracovat dodatečným programem. Pro zobrazení na počítači je ovšem jednodušší jej převést do příznivějšího formátu PDF. Nadšenci do LaTeXu, ve kterém je mimochodem psána i tato práce, jistě vědí, že PS se používá jako mezikrok při kompilaci formátu PDF. [12]

Encapsulated PostScript má na rozdíl od svého předchůdce vždy jen jednu stranu. Přestože bývá tato technologie nahrazována jinými formáty, stále se rozsáhle používá pro návrhy tiskové grafiky, například bilboardů a plakátů. Velkou nevýhodou je ovšem možnost úpravy. EPS soubory se nedají po uložení upravit, a je tedy nutné provést úpravy v původní souboru a poté je znovu uložit jako EPS. [9]

PDF (Portable Document Format)

S rozvojem digitalizace ke konci 20. století bylo potřeba vyvinout přenositelný, nezávislý a univerzální formát, který by sloužil k uchování elektronických dokumentů. V roce 1993 vynalezla společnost Adobe formát PDF v projektu dříve známém jako „The Camelot Project“. Dnes se již jedná o otevřený formát standardizovaný pod ISO 32000.

Nejedná se přímo o formát sloužící k uložení počítačové grafiky, ale díky svému velkému rozšíření k tomu bývá hojně využíván.

3.4.1 Porovnání grafických formátů

Jaký formát je tedy nejvhodnější? Odpověď není až tak jednoduchá. Neexistuje jeden univerzální formát, který by zastínil ty ostatní. Každý z formátů byl vytvořen za nějakým účelem. Je tedy nutné zhodnotit požadavky na výsledný produkt a náležitě k tomu vybrat i odpovídající formát.

Formát	Paleta RGB	Vykreslování	Transparentní	Použití
GIF	256 barev	rastrové	ne	webová grafika, loga, animace
JPEG	16777216 barev	rastrové	ne	webová grafika, fotografie
PNG	16777216 barev	rastrové	ano	webová grafika, loga
SVG	16777216 barev	vektorové	ano	webová grafika, loga
EPS	-	vektorové	ano	tisk
PDF	-	mix	ano	tisk, dokumenty

Tabulka 3.2: Porovnání grafických formátů

3.5 Diagram a grafy jako prostředky vizualizace

K vizualizaci dat se běžně používají diagramy a grafy. Tyto dva termíny občas mezi sebou bývají volně zaměňovány i přesto, že to není vždy úplně správně.

Diagramy jsou stručnou vizuální reprezentací reálných a matematických objektů sloužící ke sdělení informací o vlastnostech objektů a usnadňují řešení problému. [36] Grafy je oproti tomu možno definovat jako „grafické znázornění závislosti mezi dvěma nebo více proměnnými“. [32] V důsledku toho bývají grafy vynášeny do souřadnicového systému xy nebo xyz. Je tedy možno říci, že graf je podmnožinou diagramu a diagram je nadřazeným pojmem. V anglickém jazyce se mimo diagram a graf rozlišují ještě pojmy „plot“ a „chart“.

Spojnicový diagram

Spojnicový diagram se nejčastěji využívá pro zobrazení souvislých hodnot v závislosti na čase. Obvykle vyjadřuje osa x čas a osa y naměřené hodnoty. Spojnicový diagram je velmi podobný klasickému bodovému grafu, ale liší se především ve vnímání osy x, kdy spojnicový graf obsahuje pouze jednu osu hodnot, přičemž bodový graf obsahuje dvě. [33]

Sloupcový diagram

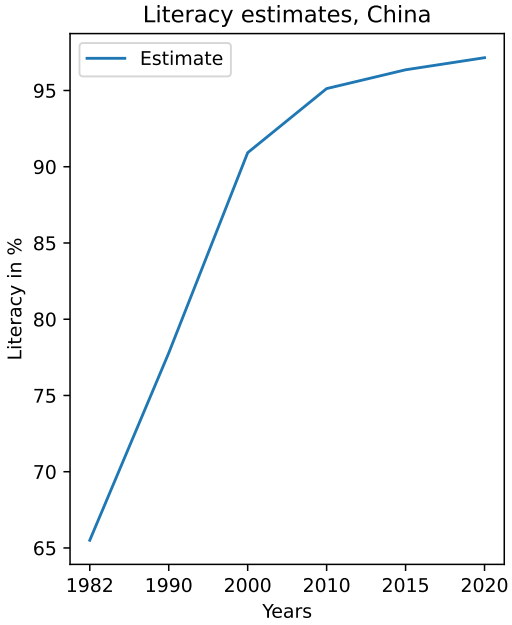
Sloupcový diagram je jedna z nejlepších možností, jak graficky reprezentovat různé skupiny dat. Takový diagram je možné znázornit jak vodorovně, tak horizontálně pomocí obdélníkových pruhů. Graf se tedy skládá ze dvou popsaných os, jejichž závislost je znázorněna

pomocí vykreslených pruhů. Sloupcové grafické znázornění se využívá především jako správa dat ve statistice nebo při porovnávání naměřených či zkoumaných hodnot.

Existuje několik různých typů sloupcových diagramů, přičemž každý takový druh je vhodný pro různou práci s daty.

- **Vertikální pruhový graf** - obvykle se hodnoty ve sloupcových grafech pohybují v závislosti na výšce sloupců (tedy na hodnotě osy y), tedy, že čím vyšší takový sloupec je, tím větší hodnotu představuje. Osa x značí jednotlivá naměřená data.
- **Horizontální pruhový graf** - je vlastně obdoba vertikálního pruhového grafu, přičemž velikost hodnot není značena vertikálně, nýbrž horizontálně.
- **Seskupený sloupcový graf** - se používá pro zobrazení diskrétních hodnot pro vícero objektů stejné kategorie. Vykresluje se pomocí principu agregace dat jednoduchého sloupcového diagramu.
- **Skládaný sloupcový graf** - se využívá pro reprezentaci více druhů dat pro jednotlivé objekty. Pro větší přehlednost se často jednotlivé části oddělují různými barvami. Nejčastěji se tímto způsobem znázorňují rozdíly porovnaných hodnot.
- **Segmentovaný sloupcový graf** - je obdoba skládaného sloupcového grafu, ale soustředí se spíše na reprezentaci vzorů či trendů v datech.
- **Dvojitý sloupcový graf** - je využíván pro reprezentaci duálních dat různé výšky.

Sloupcové diagramy představují plnou řadu výhod nejen při vizuální reprezentaci dat. Zobrazení dat je snadno srozumitelné a přímočaré. Umí také vykreslovat data, která se mění v závislosti na čase, proto je taková reprezentace hojně využívána ve velkém množství odvětví. [42]



Obrázek 3.2: Ukázka spojnicového diagramu

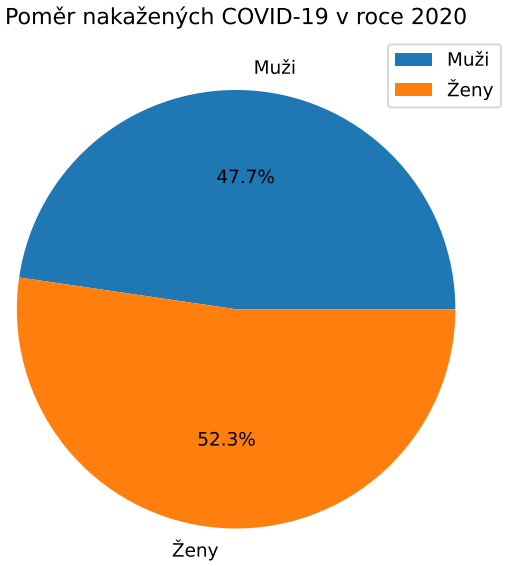
Obrázek 3.3: Ukázka vertikálního sloupcového diagramu

Výsečový diagram

Výsečový neboli koláčový diagram zobrazuje informace v kruhovém grafu a je rozdělen na jednotlivé relativní výseče. Takový graf se nejčastěji používá pro reprezentaci dat v procentech, přičemž celkový součet procent činí sto. Na základě počtu procent hodnot jednotlivých výsečí se jejich velikost přímo úměrně mění. Obvykle je také logicky uspořádaný podle velikosti hodnot. Koláčový graf se používá pro zjednodušení podání porovnaných hodnot, například poměr hodnot vůči sobě a celku nebo zdůraznění velikosti jedné hodnoty vůči hodnotě jiné.

Hodnoty je možné vykreslit různými způsoby.

- **2D výsečový graf** - je klasický kruhový koláčový graf.
- **Rozložený koláčový graf** - se tvoří rozložením 2D výsečového grafu v místě spojení jednotlivých výsečí s cílem upozornit na konkrétní oblast.
- **3D výsečový graf** - vyobrazuje data stejně jako 2D graf, ale v prostoru. [43]



Obrázek 3.4: Ukázka výsečového diagramu

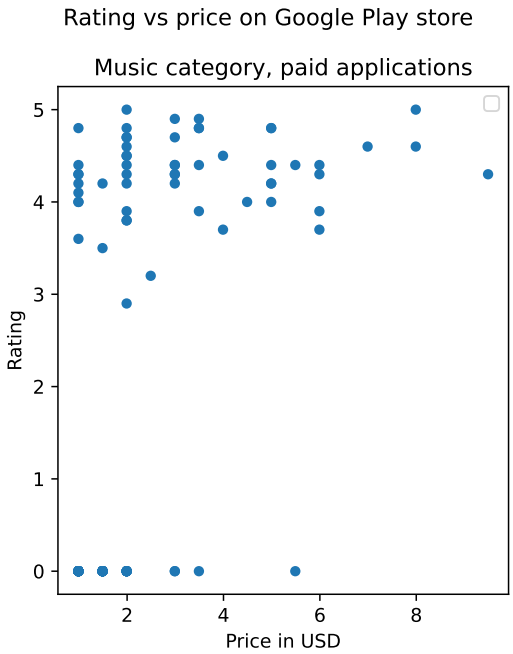
Korelační diagram

Korelační diagram se využívá pro znázornění statistické závislosti hodnot a zobrazuje jejich korelaci. Výsledné měření se značí pomocí bodů v grafu, na základě čehož pak lze zkoumat rozptyl a korelaci hodnot. Čím více jsou tedy body v grafu rozptýlené, tím méně spolu naměřené hodnoty souvisí. Směr korelace je určen místy s největší hustotou bodů. Tento druh grafu je nejčastěji využíván při analýze vztahu nějakých vstupních dat, obvykle ve statistice. Data mohou být získána z pozorování, dotazníků, průzkumů apod. Čím více hodnot je k dispozici, tím přesnější korelační graf je. Tyto grafy lze využít pro opakované porovnání naměřených hodnot (např. denně, týdně nebo měsíčně) a sledovat tak trendy v korelaci. [14]

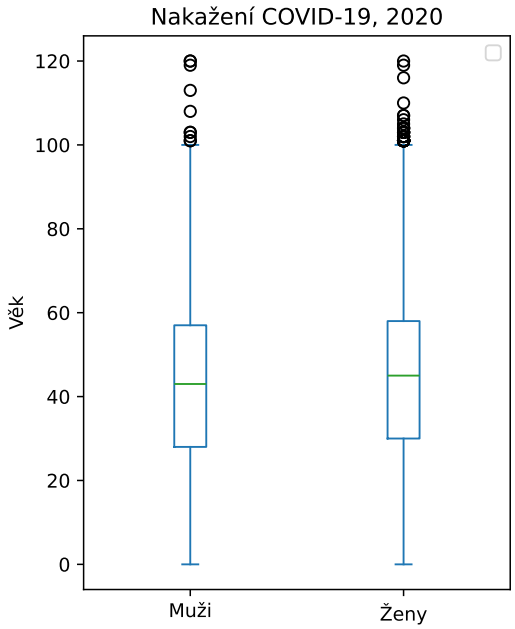
Krabicový diagram

Box-plot neboli krabicový diagram využívá tzv. kvartilů (typ statistického kvantilu) pro zobrazení dat a je využíván především ve statistice. Kvantil je taková hodnota, která rozděluje soubor seřazených (například naměřených) hodnot na několik zhruba stejně velkých částí. Kvartil je nejpoužívanější kvantil a dělí tentýž soubor na čtyři části. Dále je potřebné zmínit medián, což je prostřední hodnota (nebo průměr součtu dvou prostředních hodnot) naměřeného celku.

Krabicový diagram se tedy obvykle tvoří strojově pomocí již zažitých vzorců pro výpočet kvartilů, mediánu, kvartilového rozpětí apod. Krabicový graf se využívá především pro identifikaci odlehlé hodnoty, posouzení symetrie u konců rozdělení nebo pro porovnávání rozptylů. Takový graf se využívá převážně pro zjednodušení zobrazení hodnot statistických grafů a právě pro svoji jednoduchost se používá u automatických zkušebních a měřících zařízení, které průběžně sbírají a vyhodnocují data. [24]



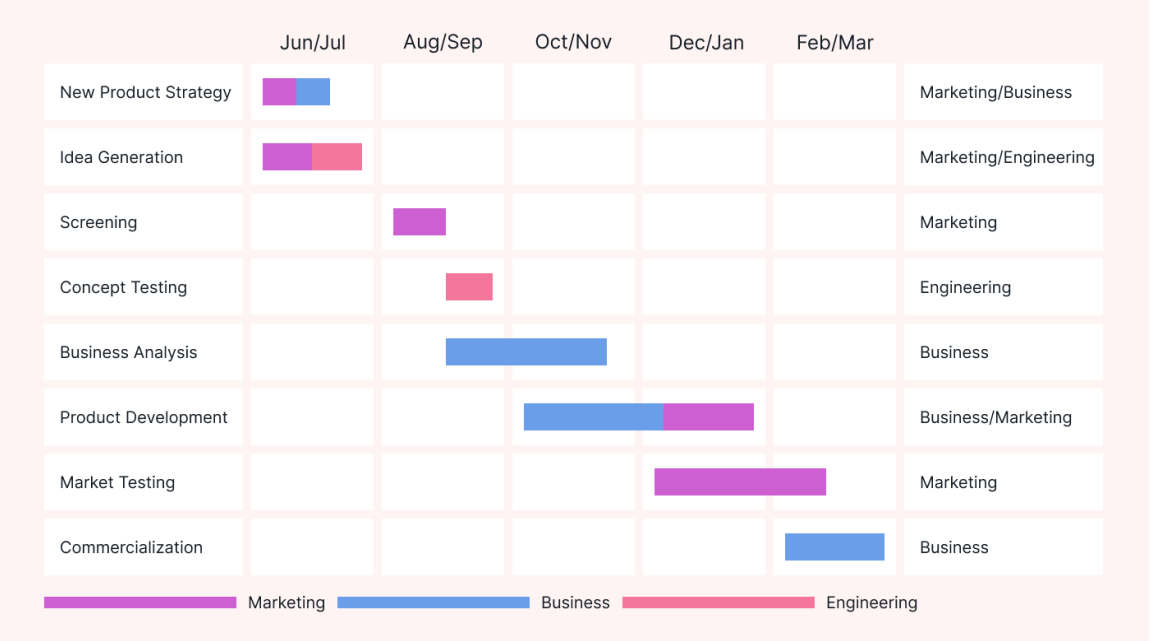
Obrázek 3.5: Ukázka korelačního diagramu



Obrázek 3.6: Ukázka krabicového diagramu

Ganttův diagram

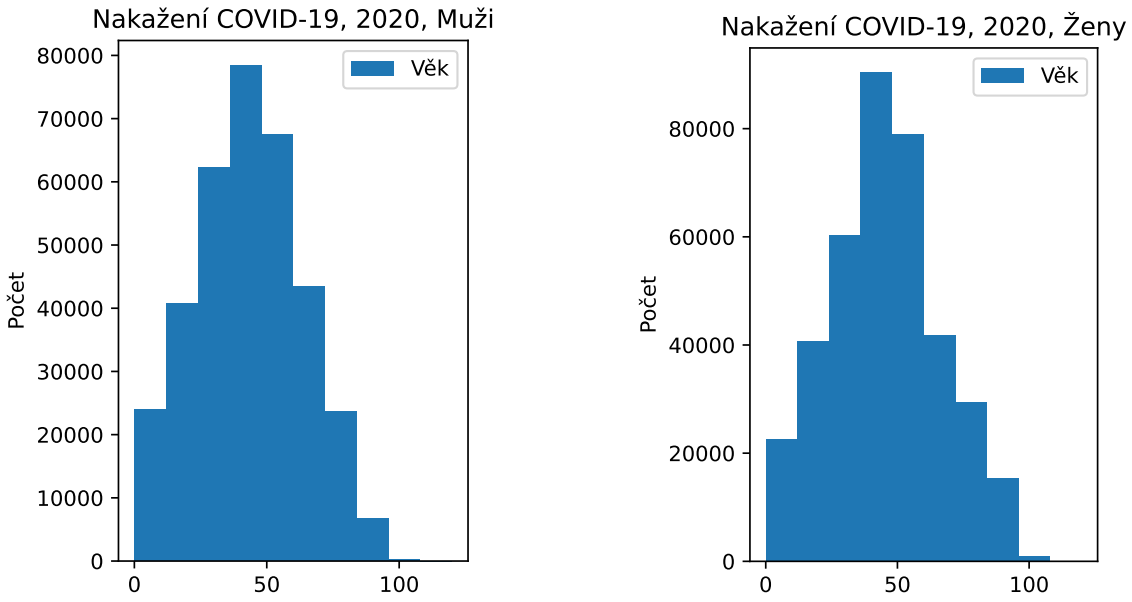
Ganttův diagram se využívá nejčastěji při prezentaci aktivit, např. v projektovém řízení. Často se tedy používá pro reprezentaci dat pro projekty těžkého průmyslu. Takový diagram se skládá z různě dlouhých vodorovných pruhů, které značí plynutí času, sekvenci úkolů, trvání a data o začátku a konci projektu. Z grafu pak lze jednoduše vyčíst, kdy se má na jakém úkolu pracovat, kdy se má úkol dokončit a v jaké části se projekt nachází. Ganttův graf dobře znázorňuje potenciální úzká místa projektu, úkoly vyloučené z časové osy nebo třeba nevyužitý či nedostatečný čas pro plnění úkolů. Ačkoliv se data v jednotlivých Ganttových grafech mění, vždy musí obsahovat seznam úkolů na ose y, průběh těchto aktivit na ose x a ukazatele průběhu pomocí vodorovných čar. [41]



Obrázek 3.7: Ukázka Ganttova diagramu. Převzato z [7]

Histogram

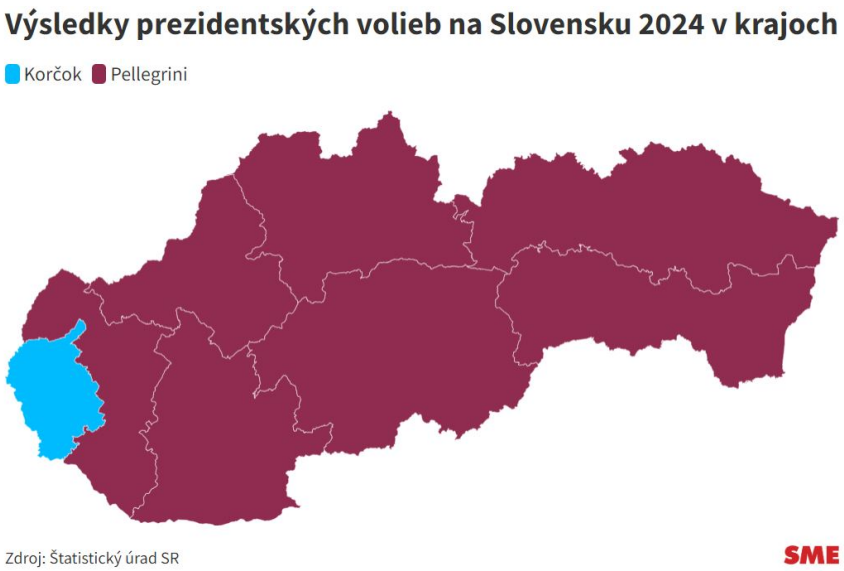
Histogram je vizuální reprezentace frekvencí pomocí sloupcového grafu pro soubor hodnot, který je rozdělen do tříd. Díky histogramu lze data posuzovat z různých hledisek, např. symetrie, vícemodálnost apod. Tvorba histogramu je závislá na několika vstupních hodnotách, a to na největším a nejmenším prvku, variačním rozpětí, počtu, hranici a šířce tříd. Histogram se také nápadně podobá krabicovému diagramu a funguje i na podobném principu. [23]



Obrázek 3.8: Ukázka histogramu

Heat map

Heat mapa neboli teplotní mapa je grafické znázornění hodnot, přičemž je každá hodnota reprezentována určitou barvou a vztahy mezi hodnotami se mohou pohybovat na určitém barevném spektru. Nejčastěji je takový graf dvourozměrný. Ve webové analýze se heat mapa používá pro sledování návštěvníka na webových stránkách, přičemž sleduje pohyb jeho kursoru a zaznamenává každé kliknutí nebo interakci. Z takové analýzy pak lze vyvodit statistiku užití všech prvků webové stránky a díky tomu lze stránku přivětivě upravit, a tím zlepšit požitek návštěvníka. [18]



Obrázek 3.9: Ukázka teplotní mapy. Převzato z [2]

3.6 Rozšířené nástroje vizualizace dat

Nástroje sloužící k vizualizaci dat nejsou rozhodně nic nového. Existuje celá řada komerčně šířených nástrojů, ať už placených či neplacených. Mezi známější nástroje sloužící čistě k vizualizaci patří například Microsoft Power BI, Tableau nebo Qlik Sense. Avšak většina uživatelů se spíše někdy setkala s nástrojem Microsoft Excel.

Komerčně šířené nástroje mají běžně rozsáhlá uživatelská rozhraní, která ulehčují práci běžným uživatelům. Uživatelské přívětivosti se dosahuje jednoduchostí používání nástroje. Grafické úpravy výstupu je možno provádět instantně, přičemž bývají nadefinovány např. předem připravené styly, mezi nimiž je možno si vybrat.

Ovšem takové nástroje mají i své nevýhody. Vstupní data dost často musí splňovat jeden z předdefinovaných formátů, aby vůbec bylo možno je zpracovat. Pokud nějaký formát nevyhovuje, je nutné použít externí nástroj, který jej převede do přijatelné podoby. Podobně mají výstupy nástroje opět předdefinovanou množinu stylů a formátů. Rozšíření takových nástrojů mnohdy není v silách typického uživatele, pokud je vůbec umožněno vytvářet či přidávat do nástroje rozšíření.

Rozhodně není záměrem komerčně šířené nástroje nějak „shazovat“. Pro většinu uživatel jsou naprosto dostačující. Ale i tak se jako vždy vyskytne někdo, kdo by rád zasahoval do fungování nástroje a upravoval jej podle svých představ.

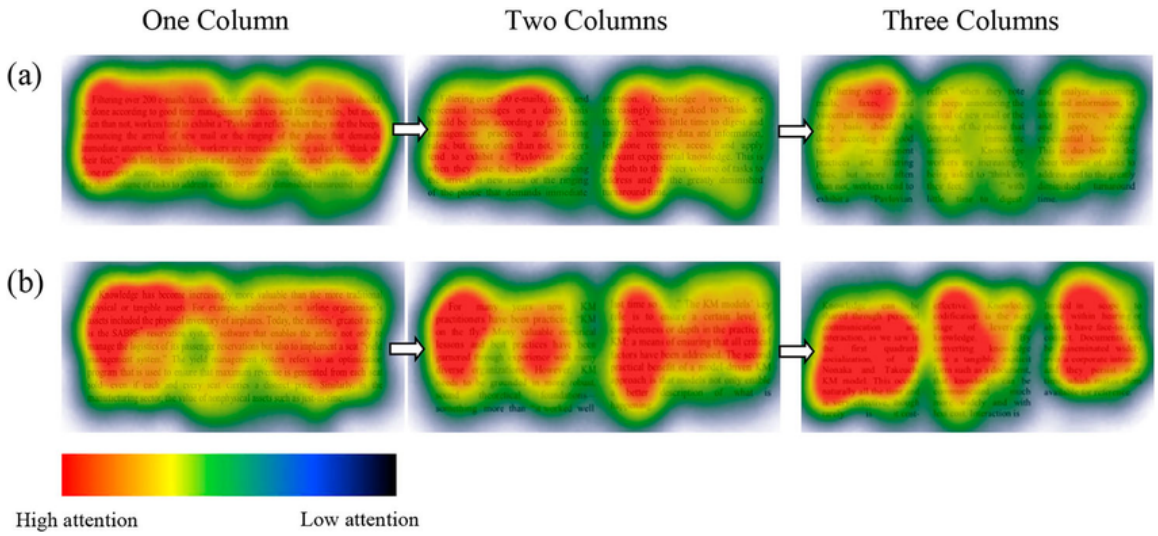
3.7 Požadavky odborných publikací na formát

Velké organizace, jako jsou například IEEE, ACM, IOP, Elsevier a mnoho dalších, umožňují profesionálům ve svých oborech publikovat odborné články, které shromažďují do sborníků. Každá z těchto organizací má vlastní požadavky na formátování a typografii článků, jež musí autoři dodržovat. Požadavky obvykle bývají obohaceny o šablony nebo vzorové texty, které slouží jako opora pro tvorbu článků. Ke tvorbě textů může být použit rozšířený nástroj Microsoft Word, ale žádanější je užití LaTeXu.

Jedním z očividných společných rysů v požadavcích je rozdělení textu do dvou sloupců. I přesto, že použití dvou sloupců nemusí být vždy vyžadováno, je lepší jej použít. Dva sloupce dávají čtenářům iluzi rychlejší četby. Dále vytvářejí efekt méně hustého a lépe rozvrženého textu.

Grafické prvky, mezi něž patří grafy a obrázky, se také řídí svými požadavky. Zpravidla se požaduje zarovnání grafických prvků na střed sloupce (v LaTeXu je upřednostňováno použití `\centering` namísto prostředí `center`). U každé grafiky bývá požadován popisek se stručným vysvětlením a číslování. Grafické prvky by neměly být seskupeny na konci článku, ale měly by se vyskytovat blízko textu, který o nich pojednává. Upřednostňovanými formáty pro vkládání obrázků jsou vektorové formáty EPS (3.4) a PDF (3.4) (kompilace LaTeXu s vloženým EPS souborem není možná pomocí nástroje `pdflatex`, proto se musí prvně provést převod vloženého souboru z EPS na PDF). Ovšem mohou být použity i rastrové formáty JPEG (3.4) a PNG (3.4) (například pro vkládání fotografií).

S výčtem jednotlivých požadavků by se mohlo pokračovat i dále, ovšem ty nejdůležitější (alespoň z hlediska zaměření této práce) byly již zmíněny. Přesto, že se jedná o odborné publikace, nároky jsou napříč organizacemi nekonzistentní. Proto je nejlepší se v případě zájmu o tvorbu odborného článku pro nějakou organizaci podívat přímo na požadavky dané organizace.



Obrázek 3.10: Tepelná mapa založená na době fixace očí na text. Řádek (a) představuje opakované čtení a řádek (b) neopakované čtení textu. Červená barva značí nejvyšší míru pozornosti, světlejší barvy poté nižší míru pozornosti. Převzato z [16]

Kapitola 4

Návrh architektury nástroje

4.1 Shrnutí požadavků

Před samotným návrhem řešení je nezbytně nutné provést shrnutí a analýzu požadavků na cílový nástroj. Požadavky už poměrně jasně vyplývají ze zadání práce, avšak je potřeba je rozebrat a objasnit.

Uživatelská přívětivost, modularita a rozšiřitelnost

Nástroj by měl být uživatelsky přívětivý, modulární a jednoduše rozšiřitelný. Uživatelskou přívětivost je možno chápat jako skupinu vlastností, které ovlivňují spokojenost uživatele. Velký vliv na spokojenost má uživatelské rozhraní, které v tomto případě může být grafické, textové nebo konzolové.

Modularita je způsob návrhu, ve kterém dochází k rozdělení programu na nezávislé zaměnitelné části (moduly). Každý z těchto modulů poté zajišťuje dílčí funkčnost. Moduly mají svoje aplikační rozhraní, která umožňují jejich spojování a navazování. Většina dnes používaných programovacích jazyků umožňuje vytváření a práci s moduly.

Nástroj by měl být uživatelem rozšiřitelný. Toho je možno dosáhnout vytvářením vlastních modulů, které mohou nahradit nebo rozšířit původní části. Vývoj modulů může být usnadněn jednoduchým, ale zároveň propracovaným aplikačním rozhraním modulů. Rozhraní musí být zároveň dobře zdokumentováno a vysvětleno.

Libovolný vstup

Nástroj by měl umožňovat načtení libovolných (textových) dat, která splňují specifikaci uživatele. Formátů pro serializaci dat je mnoho. Některé mohou být standardizovány (3.2), jiné mohou být vytvořeny přímo pro specifické potřeby uživatele atd. Vyvíjet proto nástroj, který umí pracovat pouze s hrstkou formátů, nedává úplně smysl. Zde je opět možno zmínit modularitu a rozšiřitelnost, díky které si potenciální uživatel může sám navrhnout modul pro zisk dat.

Určení podoby, způsobu a formátu výstupu

Stejně u jako každého komerčního nástroje nesmí ani zde chybět možnost, aby si uživatel určil, jak má výstup vlastně vypadat. Uživatel si musí zvolit, jaká data chce ze vstupu opravdu použít (např. vybráním sloupců). Vybraná data poté musí být možno anotovat,

zpracovat do požadovaného tvaru a vytvořit vizuální výstup. Samozřejmě i vizuální výstup má svá nastavení pro formát, rozměry, vzhled atd. Do nástroje tedy musí vstupovat uživatelská konfigurace obsahující veškeré potřebné údaje.

Generace srozumitelného výstupu

Poslední důležitý bod na seznamu požadavků je generace člověku srozumitelného výstupu z platného vstupu. Pokud tedy do nástroje vchází platná konfigurace a data jsou ve správném formátu, vypracuje nástroj prezentovatelný výstup, který bude pro uživatele srozumitelný. Ovšem zde je nutno podotknout, že pojem „srozumitelný výstup“ je velice subjektivní záležitost.

4.2 Jednotlivé části nástroje

Jelikož musí být nástroj modulární, je rozumné jej rozdělit do logických celků. Vyvstává však otázka jak? Nejlepším způsobem je uvědomit si, jakými procesy musí data projít, než jsou převedena do požadované podoby.

Načtení dat ze souboru (scanner)

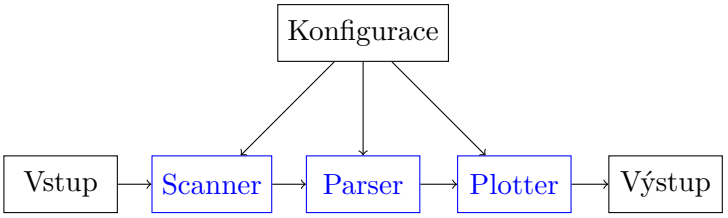
Data je před zpracováním nutno načíst. Načítání dat se musí řídit formátem uložení, aby došlo ke správné interpretaci. Pro každý formát tedy bude navržen modul pro načítání, který bude mít předepsané rozhraní.

Zpracování a úprava dat (parser)

Načtená data ve většině případů není možno přímo vizualizovat. Data proto musí být zpracována a převedena do správné podoby. Toho je možno dosáhnout například pomocí statistických operací nad daty.

Vizualizace dat (plotter)

Poslední důležitá část nástroje je modul pro tvorbu samotného vizuálního výstupu. Zde se vezmou zpracovaná data a převedou se do grafické podoby.



Obrázek 4.1: Části výsledného nástroje

4.3 Možné přístupy k implementaci

Po shrnutí požadavků a rozvržení jednotlivých částí nástroje je vhodné shrnout i možné přístupy, které lze zvolit pro samotnou implementaci.

4.3.1 Způsob nasazení aplikace

Významným faktorem při vytváření návrhu aplikace je způsob jejího finálního nasazení. Podle druhu nasazení je možno dělit aplikace na desktopové a webové. Pod pojmem desktopová aplikace se rozumí program, jenž je nainstalován na disku počítače. Pro spuštění desktopových aplikací není obvykle nutné internetové připojení, pokud se aplikace sama nepřipojuje na nějakou internetovou službu. Oproti tomu webová aplikace je dostupná pouze přes internet za pomoci prohlížeče. Připojení k internetu však nemusí být nezbytné po celou dobu běhu aplikace. Například jednostránkové aplikace (single-page applications) se při prvním spuštění stáhnou ke klientovi a pokud se nepotřebují dotazovat jiné webové služby, mohou běžet i bez připojení.

V poslední době se stává stále větší oblibou vytvářet webové aplikace. Výhod je hned několik, ale je zde opět nutné podívat se na požadavky výsledného nástroje. Pokud by aplikace běžela pro více uživatelů na internetu, byla by možnost modularity a rozšiřitelnosti pro každého z uživatelů samostatně zbytečně komplikovaná. Z toho důvodu je výhodnější zvolit jako možnost nasazení klasickou desktopovou aplikaci.

4.3.2 Volba programovacího jazyka

Volba programovacího jazyka má velký vliv na návrh vnitřního fungování jednotlivých modulů. Když už je řeč o modulech, zvolený programovací jazyk musí být modulární a bylo by vhodné, kdyby disponoval vlastností dynamického importování modulů. Většina dnes používaných jazyků však tyto podmínky splňuje. Je tedy potřeba nalézt jiné kritérium pro výběr.

Vzhledem k tomu, že nástroj musí být rozšiřitelný uživatelem, je vhodné zvolit takový jazyk, který je alespoň trochu rozšířen a používán. Existence volně dostupných knihoven by byla také velkou výhodou.

Další kritérium, které je vhodné zvážit, je způsob překladu a spuštění. Kompilované a semi-kompilované jazyky mají většinou nad interpretovanými jazyky rychlostní převahu. Ale na druhou stranu je nutno podotknout, že v tomto případě užití na rychlosti až tak moc nezáleží. Výhodou interpretovaných jazyků je, že jsou nezávislé na platformě, stačí mít jen správný interpret.

Zpracování a úprava dat v nástroji bude prováděna na základě datové analýzy, proto by bylo vhodné zvolit jazyk, který bývá k takovým případům užití běžně užíván.

Výhercem bude v tomto případě jazyk Python. Jedná se o interpretovaný, objektově orientovaný jazyk. Python je velice oblíbeným jazykem v oblastech práce s daty.

4.3.3 Možnost využití strojového učení

Stále častěji je v dnešní době možné slýchat pojmy umělá inteligence a strojové učení. Vědecké obory zkoumající umělou inteligenci se snaží plnohodnotně napodobit lidský mozek, hlavně v oblastech učení nebo řešení problémů. „Strojové učení je proces používání matematických modelů dat, pomocí kterých se počítač učí bez přímých instrukcí. Považuje se za součást umělé inteligence. Strojové učení využívá algoritmy k identifikaci vzorů v datech a tyto vzory se pak používají k vytvoření datového modelu, který dokáže formulovat předpovědi.“ [4]

I v této práci by mohlo strojové učení najít své uplatnění. Strojové učení by mohlo v jistých případech nahradit nutnost vkládání plnohodnotné uživatelské konfigurace. Bylo by

nutné například pouze určení vstupů, přičemž by nástroj sám rozpoznal vstupy, vyhodnotil nejlepší způsob zpracování a vytvořil uživateli srozumitelný výstup.

Ovšem zadání této práce klade vcelku důraz na uživatelský vstup a zásah uživatele do vzhledu či formátu výstupu. Proto i přesto, že by se jednalo o zajímavé rozšíření práce, nebude při návrhu ani implementaci počítáno s použitím strojového učení.

4.4 Uživatelská konfigurace

Jak už bylo uvedeno v požadavcích, uživateli by měla být dána možnost určit podobu a formát pro vstup a výstup nástroje. Komerčně rozšířené nástroje k tomu běžně používají grafické rozhraní, které je ideální pro běžné uživatele. Avšak například pro účely automatického testování je výhodnější možnost spuštění nástroje bez grafického rozhraní přes příkazovou řádku. Ovšem nastavování všech hodnot čistě přes příkazovou řádku není možné hned z několika důvodů. Hlavní problém je, že uživatelská nastavení netvoří konečnou množinu, protože si uživatel sám může nadefinovat vlastní hodnoty. Dalším problémem je rozsáhlost nastavení, jelikož by délka příkazu pro spuštění byla příliš dlouhá a uživatelsky nepřívětivá.

Je tedy potřeba přijít s řešením, které je možno použít v kombinaci s příkazovou řádkou, případně i s grafickým rozhraním. Jako nejjednodušší a zároveň asi i nejlepší se nabízí použití souboru s uživatelskou konfigurací. Tento soubor je možno přidat jako parametr při spuštění přes příkazovou řádku nebo může být načten přes grafické rozhraní.

Konfigurační soubor může být v jakémkoliv formátu, ale je rozumné použít nějaký ze standardizovaných formátů (3.2). Specificky k těmto účelům se velmi často používá formát JSON, který bude použit i v této práci.

4.5 Případy užití

Vzhledem k architektonickému návrhu nástroje se předpokládá existence pouze jednoho aktéra (role), kterým je samotný uživatel. Všechny možné případy užití tedy budou spojovány právě s rolí uživatele.

4.5.1 Převod vstupních dat na publikační výstup

Základním případem užití je samotný převod vstupních dat do formátu, který je vhodný použít jako publikační výstup.

Poskytnutí uživatelské konfigurace

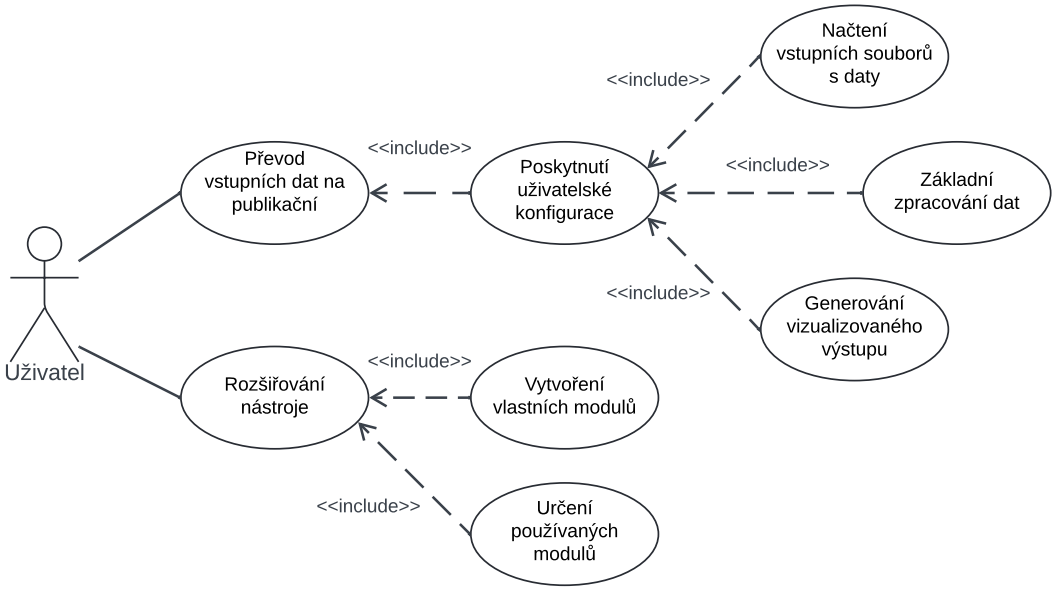
Uživatel musí mít možnost předat nástroji konfiguraci, podle níž chce, aby se nástroj řídil. V konfiguraci musí být zakomponovány informace o třech hlavních krocích nástroje.

- **Načtení vstupních souborů s daty** - Uživateli musí být umožněno vybrat si, ze kterých souborů a jakým způsobem mají být načtena vstupní data.
- **Základní zpracování dat** - Po načtení dat by měl uživatel mít možnost určit, jaká část dat má být použita a jak se mají použitá data zpracovat.
- **Generování vizualizovaného výstupu** - Na závěr procesu je nutno umožnit uživateli specifikovat podobu a formát výstupu vizualizovaných dat.

4.5.2 Rozšiřování nástroje

Vzhledem k tomu, že mezi hlavní požadavky na finální nástroj patří modularita a rozšiřitelnost, mělo by být uživateli umožněno tyto úkony provádět.

- **Vytvoření vlastních modulů**
- **Určení používaných modulů**



Obrázek 4.2: Diagram případů užití nástroje

Kapitola 5

Implementace

5.1 Základní detaily implementace

Nástroj je implementován v jazyce Python na verzi 3.10. Majoritní verze 3.10 byla vydána na podzim roku 2021. Nejedná se o nejnovější verzi jazyka Python (aktuálně je poslední stabilní verze 3.12.2), ale právě z toho důvodu byla zvolena, jelikož na novější verze nemusí existovat potřebné knihovny. Všechny zdrojové kódy nástroje jsou umístěny v adresáři `tool`.

5.1.1 Použité nestandardní knihovny

Pandas

Pandas je rozsáhlá open-source knihovna, která je používána v oblastech datové analýzy a manipulace s daty. Knihovna Pandas sama o sobě dokáže data načíst, zpracovat a vykreslit. Data je možno načíst například ze zdrojů typu `csv`, `json`, SQL databází a mnoha dalších. Pokud ve vstupu nějaká data chybí, Pandas tuto situaci automaticky vyřeší. Načtená data se ukládají do objektu typu `DataFrame`, který má zabudované indexování řádků a pojmenované sloupce. Implementace knihovny Pandas je optimalizována pro výkon, ale i přesto je velmi rychlá.

5.2 Zpracování uživatelské konfigurace

Uživatelská konfigurace je předávána pomocí konfiguračního `json` souboru, který má předepsanou strukturu odpovídající serializaci datové třídy `Config`. Základní tři sekce tvoří `inputs`, `output` a `plots`. Veškeré třídy zmíněné v této sekci jsou implementovány a umístěny v souboru `tool/models/formats.py`.

Sekce `inputs`

Vstupů může být pro jeden výstup více, proto se v této sekci nachází asociativní pole, ve kterém klíče jsou vlastní pojmenování vstupu a hodnotu tvoří serializace třídy `InputInfo`. Jediný povinný atribut třídy, který se musí vyskytovat v konfiguračním souboru, je `path` uvádějící cestu k souboru se vstupními daty. Pokud soubor podléhá standardu a má správnou příponu, je možno ponechat všechny ostatní atributy na výchozí hodnotě.

- `path` - cesta k souboru se vstupními daty.

- `format` - formát vstupního souboru. Pokud údaj chybí, je automaticky doplněn z přípony souboru.
- `delimiter` - charakter oddělující hodnoty v záznamu.
- `quoteChar` - charakter, který je možno použít pro ohraničení jedné hodnoty v záznamu.
- `newLine` - charakter označující nový řádek.
- `stripItems` - příznak, který určuje, zda mají být odstraněny prázdné znaky ze začátku a konce hodnoty záznamu.

```
{
    "inputs": {
        "group1": {
            "delimiter": ",",
            "newLine": "\n",
            "path": "file.csv",
            "quteChar": "\"",
            "stripItems": true
        },
        "group2": {
            "format": "json",
            "path": "file.json"
        }
    }, ...
}
```

Listing 4: Ukázka sekce `inputs` z uživatelské konfigurace

Sekce `output`

Z nástroje může vycházet jen jeden výstup současně, proto je sekce `output` tvořena jedinou serializací datové třídy `Output`. Stejně jako u definice vstupů, i zde je samotným povinným údajem cesta k souboru, tentokrát k výstupnímu. Zbytek opět může být ponechán na výchozí hodnotě.

- `file` - nastaví název výstupního souboru. Může, ale nemusí obsahovat příponu.
- `format` - udává výstupní formát souboru. Pokud atribut `file` obsahuje příponu, je tento údaj nepovinný a odvodí se z přípony formátu.
- `rows` - nastaví počet řádků mřížky.
- `cols` - nastaví počet sloupců mřížky.
- `rowdef` - nastaví relativní výšky řádků mřížky pomocí pole celočíselných hodnot.
- `coldef` - nastaví relativní šířky sloupců mřížky pomocí pole celočíselných hodnot.

- `header` - nastaví nadpis pro celý výstup.
- `width` - nastaví šířku celého výstupu v pixelech.
- `height` - nastaví výšku celého výstupu v pixelech.
- `dpi` - nastaví hodnotu DPI (*dots pet inch*) pro výstup. DPI udává, kolik teček (pixelů) se zobrazí na délce jednoho palce.

```
{
   "output": {
        "rows": 1,
        "cols": 3,
        "dpi": 300,
        "file": "output.pdf",
        "format": "pdf",
        "header": "Overall header",
        "height": 1080,
        "width": 1920,
        "rowdef": [1],
        "coldef": [1, 2, 1]
    }, ...
}
```

Listing 5: Ukázka sekce `output` z uživatelské konfigurace

Sekce `plots`

Poslední, a možná i nejdůležitější sekce, slouží pro určení samotného formátu a vzhledu jednotlivých vizualizací. Sekci `plots` tvoří pole serializací datové třídy `PlotInfo`. Sice se zde nevyskytují žádné povinné položky, ale je doporučeno vyplnit všechny možné hodnoty.

- `row` - určuje řádek mřížky, ve kterém bude umístěn.
- `col` - určuje sloupec mřížky, ve kterém bude umístěn.
- `source` - odkazuje na jeden z pojmenovaných vstupů. Pokud tato položka není nastavena, ve výchozím nastavení se použije první ze vstupů.
- `plottype` - značí typ grafu. Datový typ tohoto atributu tvoří třída `PlotType`, která je výčtem předem definovaných typů grafu.
- `header` - nastaví nadpis.
- `xlabel` - vytvoří popisek osy x.
- `ylabel` - vytvoří popisek osy y.
- `plotdata` - tento atribut je tvořen polem serializací datové třídy `Plotdata`. Každý objekt typu plotdata v tomto poli slouží k výběru specifického sloupce z celkového načteného datového rámce, jeho anotaci a určení typu operace nad ním.

- `col` - název sloupce (nebo klíč k hodnotě) z načtených dat, který má být použit k následnému zpracování.
- `action` - akce, nebo seznam akcí, které mají být pro vybraná data použita.
- `annotate` - nastaví vlastní popisek sloupce.
- `replace` - slouží k nahrazení původních hodnot novými. Údaje jsou uloženy jako slovník `{původní hodnota : nová hodnota}`.
- `filter` - provede vyfiltrování pouze chtěných hodnot. Hodnoty, které mají být dále použity jsou uloženy do pole.
- `sort` - provede seřazení hodnot. V základu řazení neprobíhá, ale může být nastaveno pro řazení vzestupně (`"asc"`) nebo sestupně (`"desc"`).
- `use` - určí, jestli má být vybraný sloupec použit pro výstup, nebo sloužil pouze jako prostředek filtrování či řazení.
- `dtype` - nastaví požadovaný datový typ pro hodnoty sloupce. Datový typ je možno zadat ve formátu řetězcového aliasu datových typů knihovny `Pandas` [1].

```
{
   "plots": [
        {
            "row": 1,
            "col": 1,
            "header": "Local header",
            "plottype": "barh",
            "source": "group1",
            "xlabel": "Label for x axis",
            "ylabel": "Label for y axis",
            "plotdata": [
                {
                    "col": "column1",
                    "action": "groupby",
                    "filter": [2020, 2021, 2022, 2023],
                    "sort": "asc",
                    "use": true
                }, {
                    "col": "column2",
                    "action": "max",
                    "annotate": "Data",
                    "dtype": "int"
                }
            ]
        }
    ], ...
}
```

Listing 6: Ukázka sekce `plots` z uživatelské konfigurace

[1] https://pandas.pydata.org/docs/user_guide/basics.html#basics-dtypes

5.3 Implementované pomocné třídy

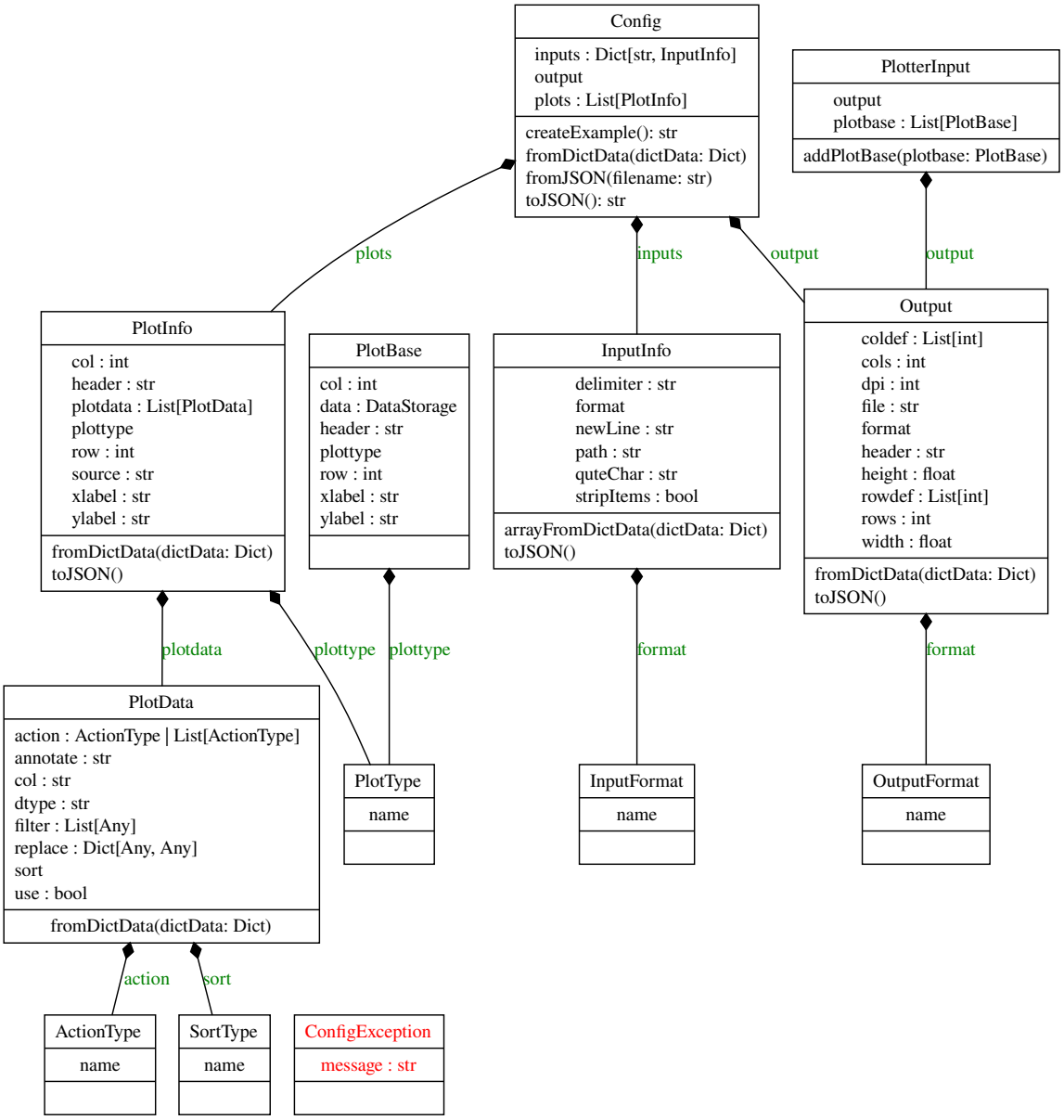
5.3.1 Třídy v modulu `tool/models/formats.py`

Některé z implementovaných tříd v modulu `tool/models/formats.py` jsou již částečně popsány v sekci 5.2, ale i přesto stojí alespoň za zmínku.

- **Config** - slouží k uchování uživatelské konfigurace. Objekty této třídy je možné vytvořit deserializací konfiguračního JSON souboru (jak již bylo zmíněno v sekci 5.2).
- **InputFormat** - třída deklarující výčet všech možných vstupních formátů, které jsou nástrojem podporovány.
- **Output** - tvoří výčet všech možných výstupních formátů podporovaných nástrojem.
- **PlotType** - deklaruje výčet podporovaných typů grafů.
- **ActionType** - udává všechny podporované statistické operace.
- **InputInfo** - uchovává informace o vstupním souboru, z nějž mají být načtena data.
- **Output** - slouží k deklaraci podoby a formátu výstupu nástroje.
- **PlotInfo** - uchovává informace o každém z jednotlivých nezávislých diagramů, z nichž se skládá jeden výstup.
- **PlotData** - ukládá informace o datech diagramu. `PlotInfo` obsahuje seznam objektů třídy `PlotData`, přičemž každý objekt obsahuje jméno sloupce ze vstupu, operace nad ním a anotaci sloupce.
- **PlotBase** - objekty třídy `PlotBase` obsahují z větší části kopii informací z objektů `PlotInfo`. Ovšem místo seznamu objektů typu `PlotData` se v objektech třídy `PlotBase` vyskytují již zpracovaná data.
- **PlotterInput** - obsahuje všechna nutná data pro moduly typu `plotter`, které pro vytvoření uživatelského výstupu potřebují specifikaci formátu výstupu a data pro vytvoření jednotlivých diagramů (nebo pod-diagramů). Specifikaci výstupu obsahuje třída `Output` (viz výše). Data pro diagramy (nebo pod-diagramy) jsou uložena v objektech seznamu, přičemž objekty seznamu jsou typu `PlotBase`.
- **ConfigException** - rozšiřuje třídu `Exception`. Výjimka typu `ConfigException` je vyvolána v případě chyby při zpracování uživatelské konfigurace.

```
$python3.10 tool/main.py -c incorrect_path
Provided config file doesn't exist
$python3.10 tool/main.py -c examples/incorrect1_config.json
Actions of type ['wrong_action'] are not supported
$python3.10 tool/main.py -c examples/incorrect2_config.json
Config json file must contain "inputs" section
```

Obrázek 5.1: Ukázka chyb zpracování uživatelské konfigurace vyvolávající výjimku `ConfigException` na snímku příkazové řádky



Obrázek 5.2: Diagram tříd v modulu `tool/models/formats.py`

5.3.2 Třídy v ostatních modulech

StartupOptions

Třída `StartupOptions` implementovaná v modulu `tool/models/options.py` slouží ke zpracování argumentů příkazové řádky, které se zadávají při spuštění programu. Možné argumenty pro spuštění jsou dále popsány v sekci 5.6.

DataStorage

Třída `DataStorage` z modulu `tool/models/storage.py` slouží převážně jako obalující třída (wrapper) nad třídou DataFrame knihovny Pandas. Třída `DataStorage` se v implementaci používá na rozhraních pro hlavní moduly programu.

5.4 Hlavní části nástroje

Nástroj má tři hlavní části, jimiž jsou scanner, parser a plotter. Každá z těchto částí je implementována samostatným modulem, který má své předepsané rozhraní. Pro každou část může současně existovat více implementací, přičemž každá z implementací odpovídá příslušné abstraktní třídě.

5.4.1 Scanner

Scanner je část sloužící k načítání dat ze vstupního souboru. Data jsou při načítání převáděna do interní reprezentace.

Moduly typu `Scanner` jsou umístěny v adresáři `scanners` a implementují abstraktní třídu `AbstractScanner`. Abstraktní třída `AbstractScanner` deklaruje abstraktní metodu `ReadFile`, která přijímá vstupní parametr typu `InputInfo` a jejímž výstupem je objekt typu `DataStorage`.

5.4.2 Parser

Parser slouží ke zpracování načtených dat. Vstupní soubory mohou obsahovat data, která jsou pro finální vizualizaci zbytečná, proto je prvním krokem zpracování právě extrakce skutečně potřebných dat. Tím ovšem proces zpracování nekončí. Dále je nutno tato extrahovaná data připravit na vizualizaci. Toho je možno dosáhnout základními statistickými metodami. Data v tomto kroku prochází seskupením, výběrem největší nebo nejmenší hodnoty, mediánu, spočítáním sumy nebo celkového počtu výskytů. O uplatněné statistické metodě rozhoduje uživatel v konfiguračním souboru (v sekci `plots` → `plotdata` → `action`).

Moduly typu `Parser` se vyskytují v adresáři `parsers` a jsou implementací abstraktní třídy `AbstractParser`. `AbstractParser` deklaruje ve své implementaci abstraktní metodu `Parse`, do níž vstupují data v objektu typu `DataStorage` a informace potřebné ke zpracování v objektu typu `PlotInfo`.

5.4.3 Plotter

Poslední důležitou částí je plotter, ve kterém dochází k samotné vizualizaci dat. Způsobů vizualizace je nespočet. Mnohé je možno vytvořit přímo pomocí knihoven pro Python, jiné vyžadují navázání na externí nastroje.

Moduly typu `Plotter` se nacházejí v adresáři `plotters` a implementují abstraktní třídu `AbstractPlotter`. `AbstractPlotter` předepisuje abstraktní metodu `Plot`, jejímž vstupem je objekt typu `PlotterInput`.

5.5 Registrace a používání výchozích modulů

Jelikož je nástroj modulární, musí existovat způsob, jakým si uživatel dokáže zvolit, který z implementovaných modulů bude používán. Aby toto bylo vůbec možné, musí se zajistit dynamické importování modulů. K tomu je v jazyce Python možné použít knihovnu `importlib`[2].

V návrhu tohoto nástroje se počítá s tím, že implementované moduly budou umístěny ve správném adresáři (balíčku). Z importovaných modulů se následně extrahují třídy, jež odpovídají požadavkům na implementaci (tzn. implementují správnou abstraktní třídu).

[2] https://docs.python.org/3/library/importlib.html

Avšak importovat implementované třídy nestačí. Dále je nutné určit, které z nich se opravdu použijí. Takové rozhodnutí provádí uživatel nástroje. Způsobů provedení je zde několik. Jednou z možností je požadovat, aby uživatel při každém spuštění nástroje provedl výběr třídy, která bude použita. Ovšem není důvod, aby nástroj při každém spuštění musel obdržet tuto informaci od uživatele. Přeci jen, přidávání nového modulu nebo jeho výměna pravděpodobně nebude prováděna při každém spuštění nástroje. Nástroj by si tedy mohl uchovávat předem nastavené hodnoty.

K implementaci tohoto mechanismu se v nástroji využívá konfiguračního souboru typu INI (s příponou `ini`). Jedná se o soubory, které umožňují uchovávat konfiguraci počítačového softwaru. Obsah INI souboru je možno rozdělit do několika sekcí, přičemž každá sekce obsahuje data uložená jako dvojice klíč–hodnota (ukázka formátu ve výpisu 8). V jazyce Python je možné pro práci s těmito soubory použít knihovnu `configparser`[3]. Nastavení výchozích tříd je dále uživateli umožněno přímo pomocí samotného nástroje. Spuštěním nástroje s argumentem `-m` nebo `--module` je možno vstoupit do režimu nastavení výchozích modulů. Po dokončení nastavení je nástroj ukončen a ostatní argumenty jsou ignorovány. Výběr za pomoci příkazové řádky je možný díky knihovně `inquirer`[4]. Přepínač `-m` nebo `--module` je možno doplnit o název sekce, pro kterou má být výběr proveden (`scanners`, `parsers`, `plotters`). Pokud není u přepínače dodáno upřesnění, pracuje nástroj automaticky v režimu nastavení všech modulů (`all`). Pokud by však někomu nevyhovovala tato metoda, je možné upravit konfigurační soubor pomocí klasické editace textu, pokud bude zachován správný formát.

Konfigurační soubor nástroje má tři sekce, jimiž jsou `SCANNERS`, `PARSERS` a `PLOTTERS`. V sekci `SCANNERS` se registrují třídy z modulů typu scanner (5.4.1), přičemž jako klíč slouží vstupní formát souboru a hodnotou je jméno implementované třídy. Obdobně je vytvořena i sekce `PLOTTERS` registrující moduly typu plotter (5.4.3), jen tentokrát je klíčem výstupní formát. Odlišně je tvořena až sekce `PARSERS`, která obsahuje pouze jeden údaj. Implementace totiž počítá pouze s jedním registrovaným modulem typu parser současně. V této sekci tedy tvoří dvojici klíč, jímž je pevný řetězec `parser`, a hodnotou je opět název implementující třídy.

Nástroj lze používat, i když pro některý ze všech možných vstupních nebo výstupních formátů nebude nastavena výchozí třída. To však platí jen v případě, že takový formát nebude vyžadován pro vstup ani výstup (jinými slovy nebude použit při běhu programu).

```
$ python3.10 main.py -m
$ python3.10 main.py --module
$ python3.10 main.py -m scanners
$ python3.10 main.py -m parsers
$ python3.10 main.py -m plotters
```

Listing 7: Ukázka spuštění nástroje v režimu nastavení výchozích modulů

[3] https://docs.python.org/3/library/configparser.html
[4] https://pypi.org/project/inquirer/

```
[SCANNERS]
csv = CSVScanner
json = JSONScanner
gantt = GanttScanner

[PARSERS]
parser = MainParser

[PLOTTERS]
pdf = PlotlyPlotter
eps = MatPlotLibPlotter
png = MatPlotLibPlotter
svg = PlotlyPlotter
tex = MatPlotLibPlotter
show = MatPlotLibPlotter
```

Listing 8: Ukázka formátu souboru `config.ini`

5.5.1 Možnosti rozšíření funkcionalit

Nástroj byl navržen tak, aby byl co nejjednodušeji rozšiřitelný a modifikovatelný. Proto může být rozšířen hned v několika směrech.

Vytváření nových modulů typu scanner, parser nebo plotter

Za předpokládaně nejběžnější rozšíření je možno považovat právě novou implementaci jednoho ze základních modulů. Pro toto rozšíření je nutno dodržet požadované předpoklady. Nový modul musí být umístěn do správného adresáře. V tomto modulu musí být umístěna třída, která implementuje předepsanou abstraktní třídu a všechny její funkce. Jsou-li tyto podmínky dodrženy, je možno takový modul začít používat, k čemuž je potřeba modul zaregistrovat (5.5).

Přidání nových vstupních/výstupních formátů

Přidání nového formátu už vyžaduje zásah do původního kódu. Nový vstupní formát je nutno přidat do výčtu `InputFormat` (`tool/models/formats.py`), aby bylo možné jej zaregistrovat. Dále bude pravděpodobně nutné vytvořit i nový modul pro čtení tohoto formátu.

Přidání výstupního formátu opět vyžaduje doplnění do správného výčtu, tentokrát `OutputFormat` (`tool/models/formats.py`). Dále může být vytvořen nový modul pro zpracování tohoto formátu, nebo může být upraven už existující modul.

Vytvoření grafického rozhraní

Aktuální implementace v sobě nezahrnuje grafické rozhraní, avšak jeho vytvoření by mohlo být realizovatelné hned několika způsoby. Základní možností je vytvořit grafické rozhraní v jazyce Python za použití jedné z již existujících knihoven (např. tkinter nebo PyQt). Takové grafické rozhraní by sloužilo k inicializaci třídy `Config` uživatelskou konfigurací.

Další možností je například vytvoření „obalujícího“ programu (tzv. wrapper), který by mohl být implementovaný v libovolném jazyce a technologii. Program by sloužil k vytvoření konfiguračního JSON souboru a spouštění nástroje s vytvořenou konfigurací.

5.6 Spuštění a práce s nástrojem

Nástroj je možno spustit z příkazové řádky. Před spuštěním je však nutno zajistit, že je v systému nainstalována správná verze interpretu a knihoven. Nástroj pro spuštění vyžaduje interpret jazyka Python na verzi 3.10. K instalaci Python knihoven slouží nástroj `pip`. Seznam použitých knihoven je dále shrnut v souboru `requirements.txt`.

Nástroj je možno spustit tak, jak je vyobrazeno níže, přičemž seznam všech možných argumentů (nahrazení za OPTION) je uveden a vysvětlen v tabulce 5.1.

```
$ python3.10 main.py {OPTION}
```

-h, --help	Vypíše pomocnou hlášku na standardní výstup a ukončí nástroj.
-m, --module	Spustí režim nastavení výchozích modulů.
-c <file>, --config <file>	Slouží pro určení souboru s uživatelskou konfigurací.
-e, --example	Na standardní výstup vypíše příklad uživatelské konfigurace.

Tabulka 5.1: Možné argumenty pro spuštění

```
$python3.10 tool/main.py -c examples/bar_config.json
$python3.10 tool/main.py -c examples/incorrect1_config.json
Actions of type ['wrong_action'] are not supported
$python3.10 tool/main.py -m parsers
Creating config for PARSER...
[?] Which will be the default parser:
 > MainParser
```

Obrázek 5.3: Ukázka práce s nástrojem

5.7 Dokumentace zdrojového kódu

Důležité části zdrojového kódu jsou opatřeny komentářem, který stručně vysvětluje implementaci. Pomocí komentářů může být vygenerována interaktivní dokumentace zdrojového kódu ve formátu `html` stránek za pomoci nástroje `pdoc` [5].

```
$ pdoc --html <modul> --output-dir <adresar>
```

[5] https://pdoc3.github.io/pdoc/

Kapitola 6

Testování a vyhodnocení

Testování výsledného produktu je stejně důležité jako návrh a samotná implementace.

6.1 Možnosti testování

Způsobů, jakými je možno testovat implementované nástroje, existuje mnoho. Každý způsob je vhodný použít k něčemu jinému a může odhalit jiné chyby. Právě proto je mnohdy žádoucí testovat programy vícero způsoby. Jak ale nejlépe otestovat implementaci v této práci? Přesná odpověď na tuto otázku pravděpodobně neexistuje. Nejvhodnější asi bude prvně zmínit některé běžně používané způsoby a z nich si vybrat.

Systémové testování

Systémové testy se zaměřují na aplikaci jako celek a nahlížejí na ni stejně, jako by k ní přistupoval konečný uživatel. Ověřuje se tedy, zda je chování celého programu v souladu s požadavky.

Funkční testování

Funkční testování je způsob testování, při kterém se dá zjistit, jestli funkčnost aplikace odpovídá očekávání a zadání. Při takovém testování je zanedbáván způsob zpracování programu a testy se soustředí na kontrolu poskytnutí správného výsledku programem, na vyhledání nedostatků nebo chyb aplikace. Funkční testování se tedy při testování softwaru zabývá tím, zda každá funkce programu odpovídá požadavkům.

Funkční testování je velmi rozsáhlý pojem a zahrnuje několik typů testování.

- **Testování jednotek** (častěji označované anglickým názvem „unit testing“) je metoda, která se používá na testování menších funkčních jednotek (částí) kódu, typicky funkcí nebo metod. Při takovém testování je nutné pokrýt co největší část kódu, a proto se také využívá při testováním řízeném vývoji (TDD - Test Driven Development).
- **Testování kouře** (nebo také „smoke test“) se používá při testování stability kritických funkcí a systému celkově. Využívá se například při ověření stability webových stránek v případě velkého množství přihlášení na stránku.
- **Testování příčetnosti** se používá v pozdější fázi testování pro kontrolu opravených částí kódu.

- **Regresní testování** je podstatné pro kontrolu již opravených částí programu a zjišťuje, zda oprava kódu nenarušila funkčnost programu a nezpůsobila nestabilitu nezávislých funkcí.
- **Integrační testování** je využíváno pro ověření toho, zda jsou jednotlivé moduly programu propojeny a fungují společně. Zkoumá se tedy převážně logika aplikace.
- **Testování zátěže** je důležité pro pochopení funkčnosti systému při očekávaném zatížení. [3]

End-to-End testy

End-to-End testování se používá v pozdější části procesu vývoje softwaru k testování funkčnosti a výkonnosti aplikace při jejím použití jako hotového produktu. Cílem takového testování je získat lepší představu o tom, jak bude produkt fungovat při použití v reálném prostředí s předpokládanou zátěží. Tento druh testování se využívá pro co nejlepší pochopení práce uživatele s produktem a snaží se tak co nejvíce předcházet potenciálním problémům uživatele s programem.

End-to-End testování zahrnuje kontrolu několika typů činností uživatele.

- **Uživatelské funkce** - jedna z prvních věcí, na kterou se End-to-End testování zaměřuje. Je to vlastně seznam všech funkcí, které v rámci programu existují a se kterými přijde koncový uživatel do styku.
- **Podmínky** - definují testy pro testery a popisují vstupní a výstupní hodnoty, kterých má program dosáhnout.
- **Testovací případy** - nejdůležitější část End-to-End testování. Je to sada akcí, které se snaží koncový uživatel v systému provádět a je při nich pozorován vývojáři. [20]

6.2 Použité prostředky k testování

Po zmínění některých běžných praktik testování již lze zvolit ty, které jsou vhodné pro otestování implementace nástroje. V průběhu vývoje nebylo použito testování řízeného vývoje, přestože by do jisté míry našlo své uplatnění. Vzhledem k návrhu nástroje však bylo nutné ověřit správnou spolupráci modulů pomocí metody integračního testování.

Ovšem nejdůležitější jsou pro ověření funkčnosti cílového nástroje End-to-End testy. Cílový nástroj musí vyhovovat samotným uživatelům, proto by jimi měl být také otestován. Z testů pak mohou vyplynout chyby implementace či nedostatky, které znemožňují naplnit dané požadavky.

6.2.1 Testovací případy

Posadit náhodného uživatele před počítač s pokynem, aby nástroj nějak otestoval, není úplně optimální přístup. Proto je vhodné připravit pro uživatele sadu úkolů, jejichž náročnost lze stupňovat, přičemž první z úkolů mohou posloužit jako seznámení se s funkčností programu. Úkoly by měly být navrženy dostatečně různorodě, aby při jejich plnění došlo k co nejrozsáhlejšímu otestování všech částí a možností programu.

Scénáře ale samy o sobě nestačí. Pro testování nástroje je nutné vybrat vhodnou skupinu uživatelů. Už ze zadání vyplývá, že cílový uživatel musí mít alespoň základní znalosti

v oblasti programovacích jazyků a práce s textovými daty. K tomu je potřeba doplnit, že nástroj nemá uživatelské rozhraní, proto je nutno k požadovaným znalostem doplnit schopnost využívat příkazové řádky ke spouštění programů a práci s nimi. Dále by bylo vhodné, i když to není požadováno, aby cílový uživatel měl zkušenosti s datovou analýzou. Pokud tyto požadavky možný uživatel naplňuje, může být zařazen do skupiny vhodné k testování nástroje.

Zadání k otestování implementace cílového nástroje této práce je poměrně jednoduché. Každý z uživatelů má k dispozici implementovaný nástroj, kapitolu o implementaci (5), automaticky vygenerovanou dokumentaci zdrojového kódu a sadu testovacích případů.

Každý z testovacích případů se skládá ze dvou souborů. První soubor obsahuje vstupní data v některém ze serializačních formátů. V druhém souboru je uložena vizualizace dat, která byla převzata z různých publikací nebo vygenerována. Cílem uživatele je pro každý z testovacích případů vytvořit konfigurační soubor, jenž převede vstupní data do formátu, který se nejvíce podobá poskytnuté předloze.

Případ	**Diagram**	**Potřebné úkony ke zvládnutí případu**
1	sloupcový	seskupení, řazení
2	sloupcový	seskupení, filtrace, řazení
3	spojnicový	seskupení, filtrace, řazení, suma, výběr více sloupců
4	histogram	počet výskytů
5	sloupcový	seskupení, filtrace, řazení, uspořádání do maticové struktury
6	koláčový	seskupení, suma
7	krabicový	nahrazení hodnot
8	schodový	seskupení, filtrace, řazení
9	sloupcový	seskupení, nahrazení hodnot, minimum, maximum, medián
10	spojnicový	2 diagramy umístěny na 1 pole, seskupení, filtrace, řazení, řazení pomocí nepoužitého sloupce, filtrace pomocí nepoužitého sloupce
11	sloupcový, histogram a koláčový	spojení případů 4, 6 a 9 s nastavením poměru šířek sloupců

Tabulka 6.1: Tabulka testovacích případů. Pro každý testovací případ jsou zmíněny úkony potřebné k jeho naplnění a jaký diagram má být výstupem.

6.2.2 Výsledky testů podle testovacích případů

Testování nástroje bylo provedeno na vzorku deseti náhodných uživatelů (dále jako subjekty). Všechny subjekty ze skupiny byly studenty Fakulty informačních technologií Vysokého učení technického v Brně a splňovaly nutné podmínky pro zařazení do testovací skupiny. Všechny subjekty zvládly úspěšně dokončit veškeré testovací případy i přesto, že potřebovaly u některých scénářů více pokusů na vytvoření správného výstupu.

Reakce subjektů na práci s nástrojem	Počet
Ze začátku je tvorba konfiguračního souboru náročná a zabere čas.	9
Uživatelské rozhraní by ulehčilo práci s nástrojem.	8
Filtrování dat by mohlo podporovat zadání pomocí rozsahu, nerovnosti, ...	8
Je jednoduché změnit výchozí moduly.	9
Po pochopení struktury konfiguračního souboru je jeho tvorba přehledná a intuitivní.	8
Mělo by být umožněno nastavit vlastní legendu.	7
Chybí možnosti volby barevného schématu.	4

Tabulka 6.2: Reakce testovacích subjektů na práci s nástrojem

Jak je možno vidět v tabulce 6.2, většina testovacích subjektů měla k nástroji podobné připomínky. Velmi častou připomínkou je složitost konfiguračního souboru, který je nutno vytvořit (nebo upravit) pro každý testovací případ. Řešením tohoto problému by podle subjektů mohlo být dodání uživatelského rozhraní.

Další častá připomínka se vztahuje k oblasti zpracování dat, přesněji k filtraci řádků podle hodnot sloupce. V nynější implementaci lze filtr zadat jen jako pole požadovaných hodnot, což pro větší množství požadovaných hodnot může být problematické a pro výběr desetinných čísel prakticky nemožné.

V oblasti samotné vizualizace dat se rovněž vyskytly připomínky. Pro většinu subjektů byl rozhodující grafický vzhled vizuálního výstupu, jako je například barevné schéma, mřížka nebo font textů.

6.3 Srovnání s podobnými existujícími nástroji

Při hodnocení kvality implementovaného nástroje však není nutno se spoléhat jen na testování. Další možností hodnocení může být srovnání s již existujícími podobnými nástroji, které byly zmíněny v sekci 3.6, specificky s nástrojem Microsoft Power BI.

Power BI je velmi oblíbený nástroj pro datovou analýzu, který umožňuje načtení data z velkého množství běžných formátů, jejich následné zpracování a vizualizaci. Oproti finálnímu nástroji této práce je Power BI ovládán čistě přes grafické rozhraní, což může být výhodou a nevýhodou současně. Za výhodu to lze považovat v oblasti grafických úprav vizualizace, jelikož si uživatel může rychle a snadno měnit například styl, velikost, umístění legendy nebo texty.

V oblasti zpracování dat má nástroj Power BI oproti implementovanému nástroji této práce velkou výhodu. Power BI podporuje například spojování tabulkových dat nebo výběr dat za použití dotazů (anglicky „query“).

Zatímco Power BI je možno nainstalovat jen na počítače s operačním systémem Microsoft Windows (v prostředí prohlížeče je nezávislý na operačním systému), implementovaný nástroj je kompletně nezávislý na platformě a vyžaduje pro svůj běh jen nainstalovaný správný interpret.

6.4 Celkové zhodnocení

Provést celkové objektivní zhodnocení implementace není vůbec jednoduchá záležitost. V oblasti implementace je těžké provést zhodnocení už jen z toho důvodu, že se počítá s alespoň základními zásahy potenciálních uživatelů do kódu. Proto připomínky ke vzhledu vizualizace jsou sice relevantní, ale zároveň uživatelem poměrně jednoduše řešitelné. Pokud se ovšem má zvážit aktuální implementace, měla by být dostačující ke splnění požadavků ze zadání.

Porovnání s jinými nástroji je opět těžké vyhodnotit. Každý nástroj byl navržen a implementován za jiným účelem a pro jinou cílovou skupinu uživatelů. Navíc k tomu na běžně rozšířených nástrojích pracují celé týmy vývojářů s větší časovou dotací, takže pokud se jim implementovaný nástroj alespoň v něčem blíží, je to možno považovat za úspěch.

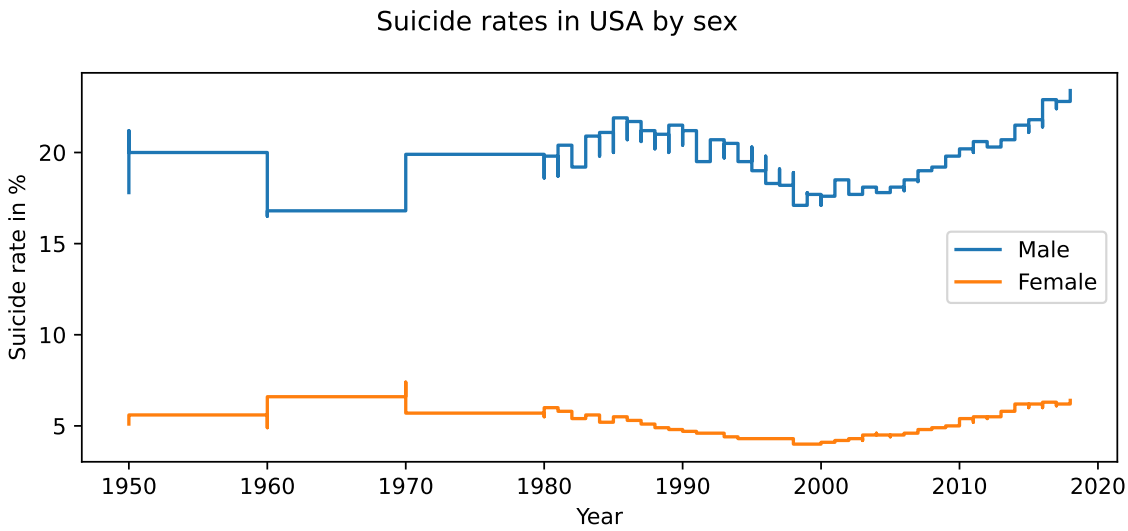
Vyvstává tu však otázka, pro koho je cílový nástroj vlastně určen? Je jednoduché stanovit požadavky na uživatele nástroje, ale ne každý, kdo tyto požadavky splňuje, bude nástroj používat. Přeci jen, běžný uživatel rozšířených nástrojů by neopustil jednoduchost grafického uživatelského rozhraní. Zároveň někdo, kdo je schopný zasahovat do připravené implementace a provádět vlastní úpravy, by měl být schopen implementovat vlastní zjednodušenou verzi nástroje, která plní jeho vlastní požadavky. I přesto je však možné, že se někdo, kdo by nástroj používal a dále rozvíjel, přeci jen najde.

6.4.1 Oblasti vhodné ke zlepšení

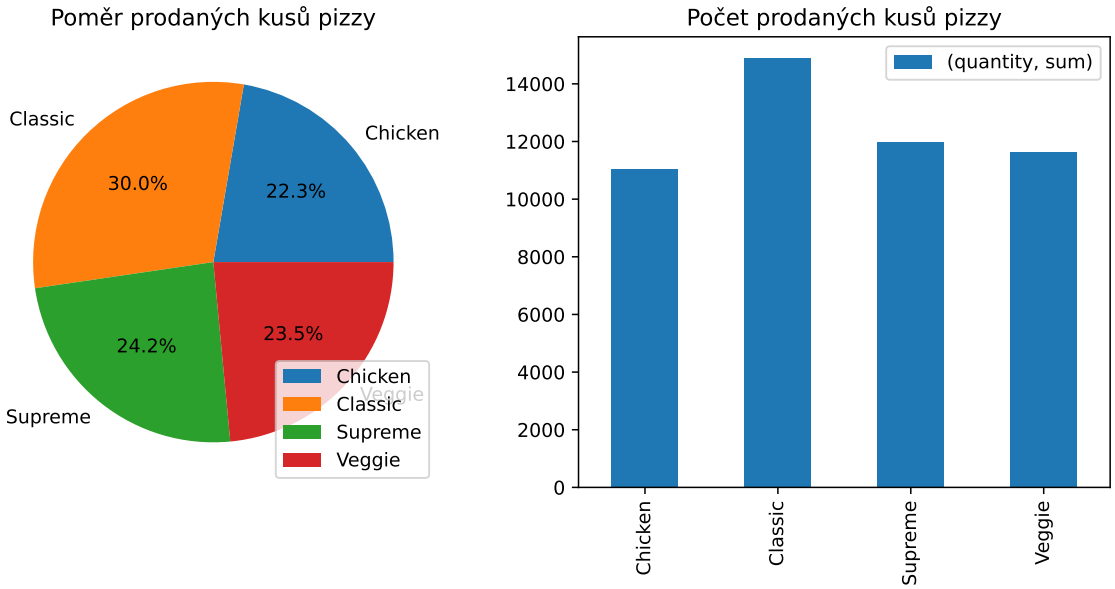
Oblasti, které by byly vhodnými kandidáty ke zlepšení či kompletní změně, vychází jak z uživatelského testování, tak z porovnání s jinými nástroji. Bylo by vhodné dopracovat moduly typu plotter, aby v základní implementaci vypadaly výstupy z nich vzhledově lépe.

Velkého zlepšení by nástroj dále dosáhl při doplnění grafického uživatelského rozhraní. Způsoby, jak toho nejlépe dosáhnout, jsou popsány již v podsekci 5.5.1.

Největším přínosem ke zlepšení by však bylo rozšíření způsobů zpracování dat. K tomu je možno zařadit již výše zmíněné možnosti pro filtrování záznamů pomocí hodnot ve sloupci.



Obrázek 6.1: Ukázka možného výstupu nástroje



Obrázek 6.2: Ukázka možného výstupu nástroje. Další zajímavé ukázky (s konfiguračním souborem, daty a vygenerovaným výstupem) jsou umístěny v adresáři `examples`.

Kapitola 7

Závěr

Cílem práce bylo navrhnout a implementovat modulární nástroj, který dokáže převést textová data na publikační výstup. Z mého pohledu byl cíl práce naplněn. Výsledkem je nástroj, který na základě uživatelské konfigurace zpracuje strukturovaná data a vizualizuje je. Vytvořený nástroj umožňuje uživateli implementovat vlastní části (moduly) nástroje, a tím jeho funkčnost rozšiřovat.

Práce se zpočátku věnovala tématu informačních zdrojů a možnostem jejich kategorizace, přičemž byla detailně rozebrána kategorie dělení podle typu nosiče. V dané části bylo provedeno podrobné dělení tištěných a elektronických pramenů, které bylo doplněno o základní informace z historie jejich vzniku. U elektronických pramenů bylo dále probráno téma důvěryhodnosti.

Dále se práce zabývala reprezentací dat a formáty uložení. Bylo zde popsáno, co je to formát souboru a jaké formáty se používají na uložení serializovaných či vizualizovaných dat. Dále byly zkoumány přístupy k vizualizaci dat, včetně možnosti využití diagramů a grafů, na což navázal průzkum v oblasti rozšířených nástrojů vizualizace dat. Na závěr kapitoly bylo provedeno shrnutí požadavků odborných publikací na formát publikovaných článků.

Praktická část práce byla uvedena návrhem architektury finálního nástroje. K tomu bylo nejprve třeba shrnout veškeré požadavky. Na základě rozboru požadavků bylo možno navrhnout základní části nástroje a vyhodnotit možné přístupy k jejich implementaci a způsobu dodání uživatelské konfigurace do nástroje. Na závěr kapitoly byly shrnuty předpokládané případy užití takového nástroje.

V implementační části byly detailně popsány základní prvky implementace, včetně použitých knihoven a technologií. Byl zde vysvětlen způsob vytvoření a zpracování uživatelské konfigurace, možnost vytvoření rozšiřujících modulů a postup jejich „registrace“ do nástroje. Popis implementace byl uzavřen popisem spuštění nástroje.

Na závěr byly zmíněny možnosti testování nástroje, z nichž bylo použito testování na uživatelích za pomoci testovacích případů. Práce končí vyhodnocením výsledků testů spolu s vlastním hodnocením implementace nástroje.

Práce mi umožnila rozvinout své zkušenosti v oblasti softwarového inženýrství. Naučil jsem se základy datové analýzy od zisku a načtení dat přes jejich zpracování a následnou vizualizaci. Dále jsem si rozšířil znalosti o informačních zdrojích a možnostech odborných publikací.

Literatura

[1] *ABOUT JPEG* [online]. [cit. 05.02.2024]. Dostupné z: https://jpeg.org/about.html.

[2] *Ako volili kraje v prezidentských voľbách 2024? (volebná mapa).* [cit. 14.04.2024]. Dostupné z: https://domov.sme.sk/c/23309905/volebna-mapa-ako-volili-kraje-prezidentske-volby-2024.html.

[3] *Co je funkční testování? Typy, příklady, kontrolní seznam a implementace* [online]. [cit. 07.04.2024]. Dostupné z: https://www.zaptest.com/cs/co-je-funkcni-testovani-typy-priklady-kontrolni-seznam-a-implementace.

[4] *Co je machine learning?* [online]. Microsoft [cit. 12.03.2024]. Dostupné z: https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-machine-learning-platform.

[5] *Dřevěný knihtiskový lis.* [cit. 23.03.2024]. Dostupné z: https://www.ntm.cz/o-muzeu/muzea-ntm/muzeum-elektrotechniky-a-medii/tiskarstvi.

[6] *File Format - What does file format mean?* [online]. Label Planet [cit. 11.02.2024]. Dostupné z: https://www.labelplanet.co.uk/glossary/file-format/.

[7] *Gantt chart example for business plans.* [cit. 14.04.2024]. Dostupné z: https://assets-global.website-files.com/62fcfcf2e1a4c21ed18b80e6/645d463a1ad4b3ed9ebb3a15_gantt_chart_example_business_planning_xrfs.png.

[8] *JPEG (JPG)* [online]. Corel Corporation [cit. 05.02.2024]. Dostupné z: https://product.corel.com/help/CorelDRAW/540240626/Main/CZ/Doc/wwhelp/wwhimpl/common/html/wwhelp.htm?context=CorelDRAW_Help&file=CorelDRAW-JPEG-JPG.html.

[9] *Soubory EPS* [online]. Adobe [cit. 06.02.2024]. Dostupné z: https://www.adobe.com/cz/creativecloud/file-types/image/vector/eps-file.html.

[10] *Soubory JPEG* [online]. Adobe [cit. 05.02.2024]. Dostupné z: https://www.adobe.com/cz/creativecloud/file-types/image/raster/jpeg-file.html.

[11] *Soubory PNG* [online]. Adobe [cit. 05.02.2024]. Dostupné z: https://www.adobe.com/cz/creativecloud/file-types/image/raster/png-file.html.

[12] *Soubory PS* [online]. Adobe [cit. 06.02.2024]. Dostupné z: https://www.adobe.com/cz/creativecloud/file-types/image/vector/ps-file.html.

[13] *Soubory SVG* [online]. Adobe [cit. 06.02.2024]. Dostupné z: https://www.adobe.com/cz/creativecloud/file-types/image/vector/svg-file.html.

[14] *Korelační diagram (Scatter diagram)* [online]. 06. července 2018 [cit. 16.02.2024]. In: ManagementMania.com. Dostupné z: https://managementmania.com/cs/korelacni-diagram-scatter-diagram.

[15] *SVG: Scalable Vector Graphics* [online]. mozilla.org, 2023 [cit. 06.02.2024]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/SVG.

[16] Al Samarraie, H., Sarsam, S. a Umar, I. *Visual perception of multi-column-layout text: insight from repeated and non-repeated reading.* Červen 2016. DOI: 10.1080/0144929X.2016.1196502. Dostupné z: https://www.researchgate.net/figure/Heat-map-based-on-fixation-duration-values-Row-a-represents-the-repeated-reading-task_fig1_304573174.

[17] ARPANET. *Arpanet map 1973.* [cit. 23.03.2024]. Dostupné z: https://en.m.wikipedia.org/wiki/File:Arpanet_map_1973.jpg.

[18] Awati, R. *Heat map (heatmap)* [online]. 2023 [cit. 16.02.2024]. Dostupné z: https://www.techtarget.com/searchbusinessanalytics/definition/heat-map.

[19] Baskar, A. *E-RESOURCES AND ITS USES IN LIBRARY* [online]. JAOURNAL of CRITICAL REVIEWS, 2017 [cit. 08.02.2024]. ISSN 2394-5125. Dostupné z: https://www.jcreview.com/admin/Uploads/Files/61c717c2ea6a74.46602997.pdf.

[20] Bose, S. *What is End To End Testing?* [online]. 20. února 2023 [cit. 07.04.2024]. Dostupné z: https://www.browserstack.com/guide/end-to-end-testing.

[21] Bray, T. *The JavaScript Object Notation (JSON) Data Interchange Format* [RFC 8259]. RFC Editor, prosinec 2017. DOI: 10.17487/RFC8259. Dostupné z: https://www.rfc-editor.org/info/rfc8259.

[22] Czaban, T. *Top 10 Proven Data Visualization Best Practices* [online]. 02. listopadu 2023 [cit. 04.02.2024]. Dostupné z: https://www.gooddata.com/blog/5-data-visualization-best-practices/.

[23] Dudek, M. *HISTOGRAM* [online]. 11. dubna 2016 [cit. 16.02.2024]. Dostupné z: https://kvalita-jednoduse.cz/histogram/.

[24] Dudek, M. *BOX-PLOT NEBOLI KRABICOVÝ GRAF* [online]. 02. října 2017 [cit. 16.02.2024]. Dostupné z: https://kvalita-jednoduse.cz/box-plot/.

[25] Erdmann, C. *Finally understanding PNG* [online]. 06. června 2021 [cit. 05.02.2024]. Dostupné z: https://compress-or-die.com/Understanding-PNG.

[26] Expert, P. T. *RASTER TO VECTOR.* [cit. 23.03.2024]. Dostupné z: https://phototouchexpert.com/service/raster-vector.

[27] Fiala, J. *Elektronické informační zdroje: využití pro život I.* [online]. Masarykova Univerzita, 2007 [cit. 05.02.2024]. Dostupné z: https://is.muni.cz/elportal/estud/ff/js07/informace/materialy/pages/eiz_opora.pdf.

[28] Fiala, J. *Kde hledat informace I* [online]. Masarykova Univerzita, 2007 [cit. 03.02.2024]. Dostupné z: https://is.muni.cz/elportal/estud/ff/js07/informace/materialy/pages/kde-hledat_opora.pdf.

[29] Hauben, M. *Historie sítě ARPANET/Internet*. Palmknihy s.r.o, 2003. ISBN 999-00-000-7834-9.

[30] Heidelbach, W. *Metal movable type*. [cit. 23.03.2024]. Dostupné z: https://commons.wikimedia.org/wiki/File:Metal_movable_type.jpg.

[31] Jandová, R. *Kniha a její podoby v 21. století*. Vysoká škola ekonomická v Praze, 2014.

[32] Kaňok, J. a Voženílek, V. Grafy a diagramy. *GeoBusiness* [online]. 2008, [cit. 16.02.2024]. Dostupné z: https://www.dibavod.cz/data/gis_kartografie/kart_grafy_a_diagramy.pdf.

[33] Lasák, P. *Spojnicový graf - Microsoft Excel* [online]. 19. září 2020 [cit. 16.02.2024]. Dostupné z: https://office.lasakovi.com/excel/grafy/spojnicovy-graf-microsoft-excel/.

[34] Machala, M. *Historie Internetu a jeho budoucí využití*. Masarykova univerzita, 2007.

[35] Machek, O. *Monografie* [online]. 03. května 2022 [cit. 03.02.2024]. Dostupné z: https://fph.vse.cz/veda/pro-zamestnance-a-doktorandy/informace-k-publikacnim-vystupum/monografie/.

[36] Manoharan, M. a Kaur, B. *Diagrams in Mathematics: What Do They Represent and What Are They Used For?* [online]. Mathematics Education Research Group of Australasia, 2022 [cit. 16.02.2024]. Dostupné z: https://files.eric.ed.gov/fulltext/ED623687.pdf.

[37] Matušík, Z. *Periodický sborník* [online]. KTD: Česká terminologická databáze knihovnictví a informační vědy (TDKIV), 2003 [cit. 13.02.2024]. Dostupné z: https://aleph.nkp.cz/F/?func=direct&doc_number=000001125&local_base=KTD.

[38] Matušík, Z. *Periodikum* [online]. KTD: Česká terminologická databáze knihovnictví a informační vědy (TDKIV), 2003 [cit. 04.02.2024]. Dostupné z: https://aleph.nkp.cz/F/?func=direct&doc_number=000001023&local_base=KTD.

[39] Matušík, Z. *Příručka* [online]. KTD: Česká terminologická databáze knihovnictví a informační vědy (TDKIV), 2003 [cit. 13.02.2024]. Dostupné z: https://aleph.nkp.cz/F/?func=direct&doc_number=000001130&local_base=KTD.

[40] Matušík, Z. *Ročenka* [online]. KTD: Česká terminologická databáze knihovnictví a informační vědy (TDKIV), 2003 [cit. 13.02.2024]. Dostupné z: https://aleph.nkp.cz/F/?func=direct&doc_number=000001138&local_base=KTD.

[41] Morales, J. *Gantt Chart: Definition, Advantage in Using it, and How to Use* [online]. 13. října 2022 [cit. 16.02.2024]. Dostupné z: https://www.mindonmap.com/blog/gantt-chart/.

[42] Morales, J. *What is A Bar Graph [Including Types and Method]* [online]. 2023 [cit. 16.02.2024]. Dostupné z: https://www.mindonmap.com/en/blog/knowledge/what-is-bar-graph/.

[43] Morales, J. *What is Pie Charting: Detailed Information about Pie Chart* [online]. 17. března 2023 [cit. 16.02.2024]. Dostupné z: https://www.mindonmap.com/blog/pie-chart-definition/.

[44] Pavlicová, L. *Elektronické zdroje. Jaké?* [online]. Ikaros, 2001 [cit. 05.02.2024]. ISSN 1212-5075. urn:nbn:cz:ik-10707. Dostupné z: https://ikaros.cz/elektronicke-zdroje-jake.

[45] Shafranovich, Y. *Common Format and MIME Type for Comma-Separated Values (CSV) Files* [RFC 4180]. RFC Editor, říjen 2005. DOI: 10.17487/RFC4180. Dostupné z: https://www.rfc-editor.org/info/rfc4180.

[46] Smetana, M. *Webová aplikace - vizualizace dat.* Vysoká škola ekonomická v Praze, 2019.

[47] Vaňura, J. *Porovnání formátů pro serializaci dat [online].* 2017 [cit. 2024-02-11]. Diplomová práce. Univerzita Hradec Králové, Fakulta informatiky a managementu Hradec Králové. SUPERVISOR: Ing. Pavel Kříž, Ph.D. Dostupné z: https://theses.cz/id/sqttfk/.

[48] Vlach, M. *Důvěryhodnost informací na Internetu* [online]. 2006 [cit. 13.02.2024]. SUPERVISOR: Antonín Rosický. Dostupné z: https://theses.cz/id/q225jy/.

[49] Voit, P. *Sborník* [online]. Knihovna AV ČR [cit. 08.02.2024]. Dostupné z: https://www.encyklopedieknihy.cz/index.php/Sborn%C3%ADk.

[50] Voit, P. *Encyklopedie knihy : starší knihtisk a příbuzné obory mezi polovinou 15. a počátkem 19. století.* 1. vyd. Praha: Libri, 2006. 1350 s. ISBN 8072773127.

[51] Voit, P. *Knihtisk 15. a 16. století* [online]. Ústav informačních studií a knihovnictví, Filozofická fakulta, Univerzita Karlova, Praha, 2008 [cit. 03.02.2024]. Dostupné z: https://sites.ff.cuni.cz/uisk/wp-content/uploads/sites/62/2016/01/Knihtisk-15.-a-16.-stolet%C3%AD_Voit.pdf.

Příloha A

Obsah přiloženého paměťového média

```
/
└── tool/
    ├── __init__.py
    ├── config.ini
    ├── reflection.py
    ├── main.py
    ├── modules.py
    ├── models/
    │   ├── formats.py
    │   ├── options.py
    │   ├── storage.py
    │   └── __init__.py
    ├── parsers/
    │   ├── mainParser.py
    │   ├── parserInterface.py
    │   └── __init__.py
    ├── plotters/
    │   ├── matPlotLibPlotter.py
    │   ├── plotlyPlotter.py
    │   ├── plotterInterface.py
    │   └── __init__.py
    ├── scanners/
    │   ├── csvScanner.py
    │   ├── ganttScanner.py
    │   ├── jsonScanner.py
    │   ├── scannerInterface.py
    │   └── __init__.py
```

```
/
├── Bakalářská_práce.pdf
├── Bakalářská_práce.zip
├── requirements.txt
├── documentation/
│   └── ...
├── examples/
│   ├── *_config.json
│   ├── *_data.(json|csv)
│   └── *_output.pdf
└── testCases/
    ├── case_*.(csv|json)
    └── case_*.(png|pdf|svg)
```

Příloha B

Návod na zprovoznění nástroje

B.1 Prerekvizity

Nástroj k správné funkčnosti vyžaduje nainstalovaný interpret **Python** na verzi **3.10** (lze použít i verzi 3.11). Dále je potřeba nainstalovat knihovny ze souboru `requirements.txt` pomocí modulu pip.

B.2 Instalace

V případě tohoto nástroje se počítá s tím, že do jeho kódu bude uživatel zasahovat, proto by také měl mít plnou kontrolu nad tím, kam a jakým způsobem se nástroj nainstaluje. Nástroj je samozřejmě možné používat i bez instalace spuštěním `main.py`, ale k tomu je nutné být v adresáři `tool` nebo znát cestu k souboru `main.py`.

B.2.1 Doporučený postup instalace

Linux

1. Zkopírujte složku `tool` do požadovaného adresáře.
2. Vytvořte bash skript soubor, který obsahuje:

   ```
   #!/bin/sh
   <cesta_k_python3.10> <cesta_k_main.py> "$@"
   ```

3. Upravte cesty ve vytvořeném souboru tak, aby odpovídaly skutečnosti. Je doporučeno použít absolutní cesty.
4. Upravte práva k vytvořenému souboru tak, aby bylo možno jej spustit.

   ```
   $ chmod +x <soubor>
   ```

5. Přidejte vytvořený soubor do `PATH`. Toho lze dosáhnout například vytvořením odkazu, který je umístěn do adresáře `/usr/local/bin`.
6. Nástroj je připraven k používání. K ověření funkčnosti je možno zkusit spustit nástroj s přepínačem `--help`.

Windows

1. Zkopírujte složku `tool` do požadovaného adresáře.
2. Vytvořte batch skript soubor (přípona `.bat`), který obsahuje:

   ```
   @echo off
   <cesta_k_python3.10> <cesta_k_main.py> %*
   ```

3. Upravte cesty ve vytvořeném souboru tak, aby odpovídaly skutečnosti. Je doporučeno použít absolutní cesty.
4. Přidejte vytvořený soubor do `PATH`. Toho lze dosáhnout například umístěním skriptu do prázdného adresáře a přidáním tohoto adresáře do `proměnných prostředí`.
5. Nástroj je připraven k používání. K ověření funkčnosti je možno zkusit spustit nástroj s přepínačem `--help`.

Příloha C

Příklady konfigurací nástroje

```
{
    "inputs": {
        "group1":{"path": "examples/pie_and_hist_and_box_data.csv"}
    },
    "output": {
        "file": "examples/pie_output.pdf",
        "format": "pdf",
        "width": 720,
        "height": 720
    },
    "plots": [
        {
            "header": "Poměr nakažených COVID-19 v roce 2020",
            "plottype": "pie",
            "source": "group1",
            "plotdata": [
                {
                    "col": "pohlavi",
                    "action": "groupby",
                    "annotate": "Pohlaví",
                    "replace": {"M": "Muži", "Z": "Ženy"}
                },
                {
                    "col": "vek",
                    "action": "count"
                }
            ]
        }
    ]
}
```

Listing 9: Ukázka konfigurace ze souboru `examples/pie_config.json`. Výstupem této konfigurace je výsečový diagram na obrázku 3.4.

```
{
  "inputs": {
    "group1": {
      "path": "examples/bar_data.json"
    }
  },
  "output": {
    "file": "examples/bar_output.pdf",
    "format": "pdf",
    "width": 800,
    "height": 1000,
    "dpi": 200
  },
  "plots": [
    {
      "header": "Experience level vs salary",
      "source": "group1",
      "plottype": "bar",
      "xlabel": "Experience level",
      "ylabel": "Money in USD",
      "plotdata": [
        {
          "col": "experience_level",
          "action": "groupby",
          "annotate": "Experience level",
          "replace": {"SE": "Senior", "EN": "Entry", "MI": "Mid"}
        },
        {
          "col": "salary",
          "action": ["mean", "median", "min"],
          "annotate": "SALARY"
        }
      ]
    }
  ]
}
```

Listing 10: Ukázka konfigurace ze souboru `examples/bar_config.json`. Výstupem této konfigurace je sloupcový diagram na obrázku 3.3.