Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Zhodnocení nástrojů pro tvorbu grafů ve webovém prostředí

Plzeň 2023 Jakub Hrušovský

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub HRUŠOVSKÝ**
Osobní číslo:	**A20B0596P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační systémy**
Téma práce:	**Zhodnocení nástrojů pro tvorbu grafů ve webovém prostředí**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Nastudujte současné přístupy k efektivní prezentaci dat.
2. Sestavte množinu funkčních požadavků na vizualizační nástroje dle poznatků z prvního bodu.
3. Navrhněte sadu modelových vizualizací a ukázková data k posouzení možností vizualizačních nástrojů.
4. Vyberte několik moderních nástrojů pro tvorbu webových vizualizací a implementujte v nich sadu modelových vizualizací.
5. Zhodnoťte jednotlivé nástroje s důrazem na vizualizační možnosti a obtížnost práce s nástrojem.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce

Vedoucí bakalářské práce: **Ing. Martin Kryl**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **3. října 2022**
Termín odevzdání bakalářské práce: **4. května 2023**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 25. října 2022

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 21. června 2023

Jakub Hrušovský

Poděkování

Děkuji panu Ing. Martinu Krylovi za odborné vedení, trpělivost a cenné rady při zpracování této práce.

Abstract

Comparison of web-based charting libraries. The aim of this work is to analyze and compare trending JavaScript libraries for visualization. First, the fundamental theoretical concepts for effective visualization are explained, then the set of selected functionalities has been defined. As part of the practical part, this work implemented selected useful functionalities for customizing the visualization from the previous point and compared each libraries based on them.

Abstrakt

Cílem této práce je analyzovat a porovnat populární JavaScriptové knihovny pro vizualizaci grafů. Jako první jsou vysvětleny základní koncepty efektivních vizualizací, za čímž následuje definice množiny funkcionalit, které byly zkoušeny. V praktické části se tato práce zabývá implementací množiny těchto funkcionalit v jednotlivých knihovnách, které následně porovnává.

Obsah

1 Úvod

Žijeme v novém věku, v 21. století, kde je každým dnem o nás shromažďováno neuvěřitelné množství dat, a to jak v zaměstnání, tak i v soukromí. Ohlédneme-li se zpět do minulosti, zjistíme, že za posledních několik let se naše společnost transformovala do podoby, kde žít bez chytrých příslušenstvích, jako jsou chytré hodinky nebo mobil, si většina z nás ani nedokáže představit. S příchodem čtvrté industriální revoluce, neboli Průmyslu 4.0, se shromažďované množství dat stále zvyšuje takřka exponenciálním tempem a v nejbližší době se nejspíše jen tak nezastaví. Je tedy zcela logické, že firmy chtějí tato nashromážděná data zpracovat a následně informace z nich využít pro monitorování a zefektivnění naší každodenní práce nejen na pracovišti, ale i v domácích podmínkách.

Palčivým problémem této problematiky je ale právě správné zpracování těchto dat. Vzhledem k obrovskému množství zpracovávaných dat je potřeba nejprve tato data nějakým způsobem pročistit od nadbytečných informací, které pro daný účel zpracování postrádají smysl, a následně zbytek začít analyzovat. Bohužel proces filtrace dat je značně komplikovaný a z toho důvodu zredukovaná data často obsahují mnoho dalších informací, které sice nemají pro daného jednotlivce nebo skupinu význam, ale nejdou jednoduše vyfiltrovat.

Hlavním cílem této bakalářské práce je předvést možnosti vizualizace dat pomocí vybraných javascriptových knihoven. V druhé kapitole pojednává bakalářská práce o základních principech správné vizualizace souboru dat, kde jsou přiloženy i příklady zmiňovaných správných i nesprávných vizualizací. Hodnocení možností vizualizace v jednotlivých knihovnách se odvíjí od několika faktorů, které byly vybrány v rámci bakalářské práce. Mezi tyto faktory patří například intuitivnost, portfolio základních funkcí, pokročilé možnosti a další, kterými se tato práce zaobírá ve třetí kapitole. Následně v kapitole čtvrté je vytvořen přehled funkcí jednotlivých knihoven a jejich porovnání. V poslední kapitole je ukázána implementace sady modelových vizualizací v těchto knihovnách. Tato bakalářská práce tedy analyzuje možnosti vizualizace nad souborem předem definovaných knihoven. Po této analýze následuje závěr a finální zhodnocení všech těchto nástrojů.

2 Obecná problematika

Vizualizace dat je klíčovým nástrojem v analytickém procesu, který umožňuje zobrazit a interpretovat data pozorovatelům, a to v jednoduché a přehledné formě. Správná vizualizace dat také může pomoci odhalit vzorce a trendy, které by mohly být přehlédnuté při pouhém čtení tabulek nebo textových výstupů a tak například poukázat na nejistý vývoj v akciových trzích.

Jak poukázala C. Knaflic [1], v dnešní době jsou vzdělávací systémy postavené tak, že kladou důraz především na výuku jazyků a matematiky. V jazycích nás učí, jak vyprávět příběhy pomocí slov, jak správně tato slova skládat a co vlastně znamenají. V matematice nás naopak učí, jak se sčítá, odčítá, násobí, dělí, učí nás římské číslice, integrály, derivace a mnoho dalšího. Bohužel ale jen málokdy jsou nám předávány znalosti, které nám tyto dvě v podstatě nezávislé entity spojí a vytvoří z nich jednotný příběh. Tomuto příběhu se v oblasti vizualizací dat říká kontext, který vlastně k jednotlivým číslům přiřazuje i jejich význam a tak vypráví jejich příběh. Jeden ze základních problémů efektivní vizualizace dat je tedy nedostatečné vzdělávání.

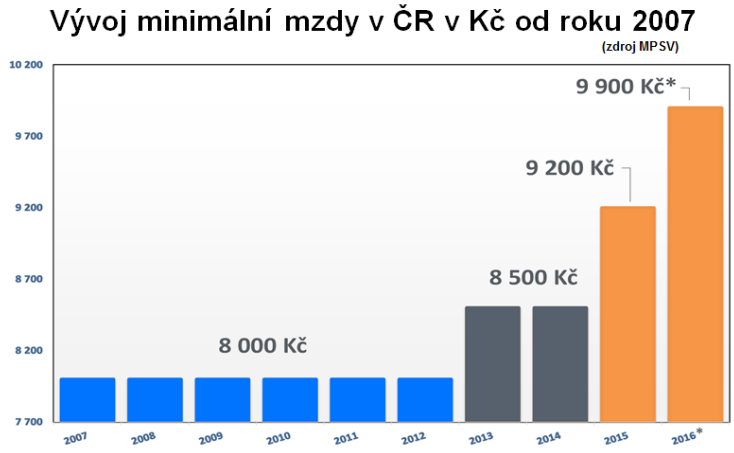
Nedostatečné vzdělání v oblasti vizualizace dat povětšinou vede k problémům, kdy lidé nejsou schopni efektivně využívat informace, které jim byly poskytnuty v grafech a diagramech, což značně ztěžuje jejich práci. Může to také vést až k tomu, že budou špatně interpretovat informace nebo budou vyvozovat mylné závěry, které mohou mít pozdější dopad na jejich práci nebo i dokonce na životní situaci. Jako příklad si můžeme uvést příběh nově zaměstnaného manažera podniku, který na toto místo nastoupil po odchodu jeho bývalého nadřízeného. Podnik si nechal od analytiků zpracovat analýzu produktivity a výsledky této analýzy byly předvedeny tomuto manažerovi. Ten následně na základě těchto výsledků propustil desítky zaměstnanců z oddělení, která se mu podle analýzy zdála jako nejvíce problémová. Avšak po několika měsících se zjistilo, že dotyčný člověk si špatně interpretoval výsledky této analýzy. Skutečně se ukázalo, že tato oddělení byla jedna z nejproduktivnějších a jejich zaměstnanci přinášeli podniku nejvíce zisku. Bohužel bylo již pozdě a podnik utrpěl velké ztráty. Propuštění zaměstnanců a snížení rozpočtu těchto oddělení mělo fatální následky pro celý podnik. Manažer byl nucen odstoupit a společnost musela znovu budovat svou reputaci, což už se ale nikdy nepodařilo.

Je tedy důležité, aby lidé, kteří pracují s daty, měli dobré znalosti o tom, jak efektivně vizualizovat sesbíraná data, aby se z nich dalo získat co možná nejvíce informací, a to takovou formou, že budou jednoznačné a přehledné. To zahrnuje znalost různých typů grafů nebo diagramů a na tomto základě mít přehled, jaké grafy nebo diagramy je vhodné použít pro daná zkoumaná data. Tedy velmi cennou znalostí není tvorba grafu, ale tvorba ideálního grafu, aby byl jednoduše pochopitelný a čitelný pro cílovou skupinu. Tato znalost tedy zahrnuje činnosti jako jaký graf vybrat, jakou legendu, barvy a mnoho dalšího. V neposlední řadě je součástí také znalost toho, jak správně interpretovat výsledky, jelikož na mnoha pozicích po celém světě lidé mají málo času a co je tedy opravdu zajímá, jsou výsledky, které pro ně hrají klíčovou roli. To především zahrnuje schopnost rozpoznat, co grafy a diagramy říkají, a vyvozovat z nich správné závěry. Je také důležité zvážit možná omezení nebo chyby, které mohou ovlivnit tyto výsledky, a brát je v úvahu při konečné interpretaci dat.

2.1 Zkreslování dat

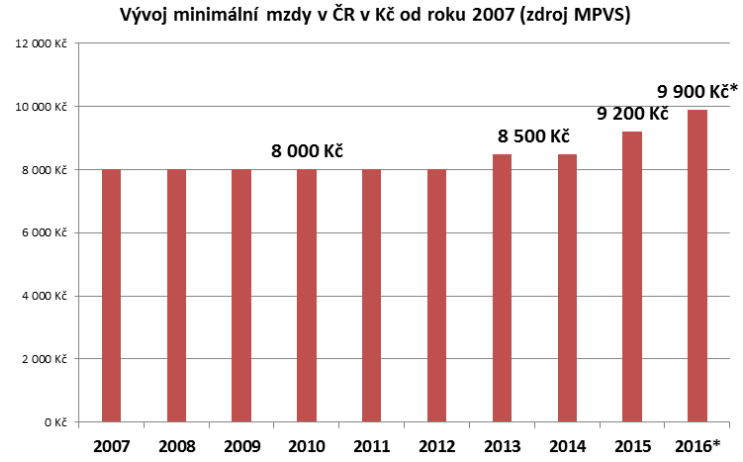
Mezi další problémy ohledně vizualizace dat patří úmyslné zkreslování dat, které je založeno na záměrné manipulaci s fakty a které vede k nesprávným rozhodnutím. Zkreslování dat je technika, která se využívá ve vizualizacích, aby se zdůraznily určité aspekty těchto dat. Zkreslování dat samo o sobě není přímo problémem, ale špatné nebo neetické využívání tohoto zkreslování již problémem je. Je tedy velice důležité si uvědomit, zda chtěné zkreslení doopravdy pomůže porozumět vašim datům, a nebo jestli se vámi předávané informace stanou chybnými a nebo zavádějícími. 2.1.

Na obrázku 2.1 lze vidět zavádějící využití zkreslování dat, kde osa y nezačíná na 0, ale začíná na 7 700. Následné navýšení částky tedy vypadá, jakoby růst částky byl více než dvojnásobný. Po podrobnější analýze ale zjistíte, že reálné zvýšení se pohybuje kolem +- 7% místo již zmiňovaného dvojnásobku. Toto je jedna z velmi často používaných praktik pro předvedení práce například za jedno volební období, kdy na první pohled se zdá, že členové vlády udělali razantní krok, kde ale po důkladnějším pohledu zjistíte, že jsou to pouze pokřivené informace.



Obrázek 2.1: Zkreslený graf vývoje minimálních mezd v ČR [1]

Nezkreslený graf lze vidět na obrázku 2.2, kde poměr lze přímo vidět a tedy navýšení o určité částky nejsou tak razantně zvýrazněné a poměry jsou blíže k realitě, než jak tomu bylo na předchozím obrázku 2.1.

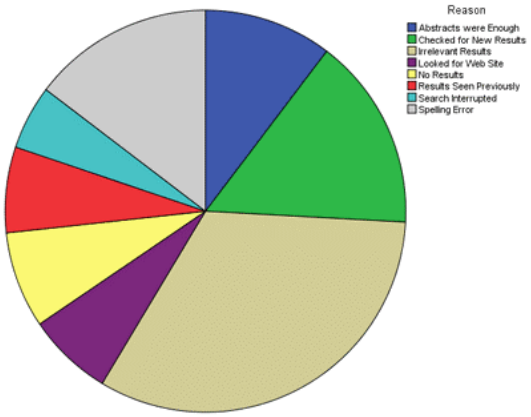


Obrázek 2.2: Nezkreslený graf vývoje minimálních mezd v ČR [2]

Zkreslování dat se ale bohužel netýká pouze sloupcových grafů, protože daleko hůře jsou na tom grafy výsečové, tzv. koláčové. Podle uznávaného amerického statistika, Edwarda Tufteho [2], by se tyto grafy neměly vůbec používat vzhledem k tomu, že jsou velice zavádějící a lehce dokážou zkreslovat realitu. Jedním z klíčových problémů koláčového grafu je, že hodnoty tohoto grafu se špatně porovnávají a pokud je informací v tomto grafu ještě

[1] Zdroj: https://1gr.cz/fotky/idnes/15/103/org/PKA5ecb9e_tweetgraf1.png
[2] Zdroj: https://1gr.cz/fotky/idnes/15/103/org/PKA5ecb9d_23_10_201520_48_21.png

více, tak je skoro nemožné přesně určit jeho hodnoty. Příklad takového koláčového grafu lze najít na obrázku 2.3



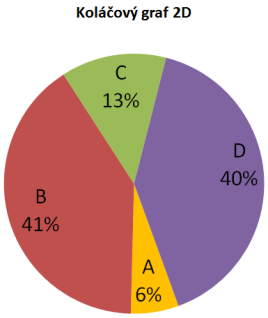
Obrázek 2.3: 2D koláčový graf [1]

K této problematice také spadá problém tzv. perspektivního zkreslení. Perspektiva jako taková je optický jev, který způsobuje, že vzdálené objekty se jeví menší než objekty blízké. To zapříčiňuje, že i když má vzdálený objekt reálně větší plochu než objekt blízký, tak se pro lidské oko jeví jako menší. Na tomto principu je tedy postavený problém perspektivního zkreslení u 3D grafů, který 2D grafy nemají. Při samotné transformaci z 3D prostoru na 2D prostor totiž dochází k úbytku informací, jelikož třetí rozměr není možný v 2D prostoru přesně vyobrazit a tím pádem zanáší do finálního obrazu již zmiňované perspektivní zkreslení. Toto zkreslení je využíváno například v herních enginech, které se snaží výsledný obraz nasimulovat tak, aby byl pro lidské oko co možná nejvíce podobný obrazu z reálného světa.

O něco hůře jsou tedy na tom 3D koláčové grafy, kde kromě špatné čitelnosti hodnot je zde vidět i poměrně zásadní zkreslení díky různým náklonům, stínům a smísení jednotlivých barev. V takovémto grafu nám může připadat, že jeden dílek je o dost větší než dílek jiný, ale může to být pouze mylný pohled na věc, zapříčiněný například náklonem takového grafu, protože ve skutečnosti druhý dílek může být menší, stejně velký a nebo dokonce o mnoho větší než dílek, se kterým ho porovnáváme. Tento problém lze vidět na obrázku 2.4 a porovnat ho se stejným grafem ve 2D na obrázku 2.5, kde lze z tohoto příkladu jednoznačně vyvodit, že použití 3D koláčového grafu je nevhodné a velmi zavádějící.

[1] Zdroj: https://wikisofia.cz/w/index.php?curid=839

Obrázek 2.4: 3D koláčový graf [1]



Obrázek 2.5: 2D koláčový graf [2]

Jakožto další poměrně zásadní problém můžeme považovat nadbytek informací a nedostatečné přizpůsobení grafů nebo diagramů svému cílovému publiku. Grafy a diagramy, jakožto nástroje pro zobrazení a interpretaci dat, by měly být navrženy tak, aby byly co nejjasnější a nejsrozumitelnější pro cílovou skupinu. Když je tedy v grafu zobrazeno až příliš mnoho informací, může to být pro člověka těžké na pochopení a následné vyhodnocení, což může vést k mylnému pochopení nebo ke ztrátě důležitých informací. Proto je velice důležité si vybrat, jaké informace mají pro danou cílovou skupinu význam a jaké je již možné zanedbat. Je tedy potřeba vzít v úvahu úroveň znalostí a schopností cílové skupiny ohledně interpretace grafů, o čemž jsme již hovořili v kapitole 2.

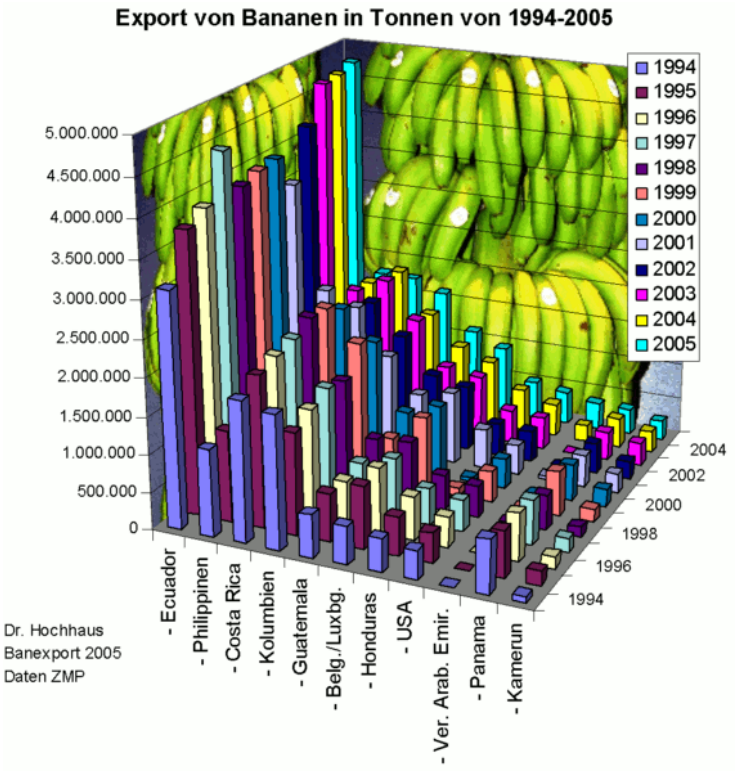
Pokud budeme například tvořit graf pro tzv. "laickou" veřejnost, měl by být navržen takovým způsobem, že nepůjde do složitých detailů dané problematiky. Graf bude tedy zaměřen na obecné informace, které by pro tuto cílovou skupinu mohly být zajímavé. Neměli bychom také opomenout, že graf nebo diagram pro takovouto skupinu je lepší skládat z jednoduchých typů grafů a diagramů a měly by přehledně zobrazovat předávané informace.

Naopak pokud je graf nebo diagram určen pro odbornou skupinu, může být vhodné použít složitější struktury grafů nebo diagramů a zobrazit k nim více detailů. Toho lze dosáhnout například rozdělením informací do několika grafů nebo diagramů a nebo přidáním komplexnější legendy.

Obrázek 2.6 ilustruje, jak nadměrné množství informací a nevhodně zvolený typ grafu dokáže ovlivnit jeho vypovídající hodnotu. Problémy jsou zde hlavně ve velmi špatně zvoleném barevném rozvržení, kde místo jednoznačně

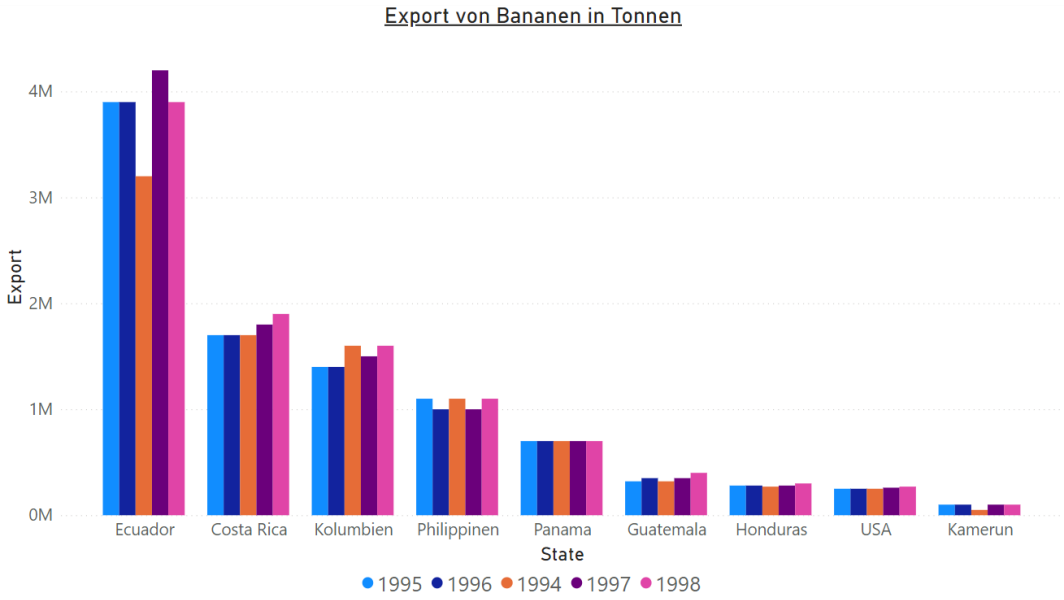
[1] Zdroj: https://1gr.cz/fotky/idnes/15/103/org/PKA5ecf73_kolac3D.png
[2] Zdroj: https://1gr.cz/fotky/idnes/15/103/org/PKA5ecf71_kolac2D.png

od sebe rozlišitelných barev byly zvoleny barvy splývající s dodatečným prvkem stínování, který ještě více napomáhá k mísení jednotlivých sloupců dohromady. Pro většinu lidí tento graf bude velmi nepřehledný a chaotický, což v důsledku znamená, že důležité informace v tomto grafu nejspíše zůstanou opomenuty a tím pádem pozorovatel může nabýt mylných informací. Lepší variantou vizualizace těchto informací by byl například rozklad na grafy o menších celcích, kde by jednotlivé informace byly od sebe jednoznačně odděleny.



Obrázek 2.6: Nepřehledný graf s nadměrným množstvím informací [1]

[1]Zdroj: https://commons.wikimedia.org/wiki/File:K%C3%BChlschifffahrt_Bananen_Exporte_2007.png



Obrázek 2.7: Příklad přehlednější zjednodušené vizualizace obrázku 2.6

Na obrázku 2.7 je vidět, jak pouhá změna grafu z 3D na 2D vizualizaci zapříčiní, že jednotlivé hodnoty grafu jsou daleko lépe čitelné, než tomu bylo v případě grafu 2.6. K tomuto efektu přispělo odebrání nadbytečných/-duplicitních popisků, konkrétně roků, a vhodnější volba barevných odstínů pro jednotlivé sloupce grafu a jeho pozadí. Právě změna barevného spektra jednotlivých sloupců k jednoznačně od sebe odlišitelných odstínů barev a zneutralizování pozadí vedla k výrazně lepší čitelnosti celého grafu oproti jeho předchůdci.

V neposlední řadě mezi problémy můžeme zařadit výběr špatných nástrojů. Dnešní doba nám na trh přinesla mnoho nástrojů, které jsou schopny zpracovávat data a následně z nich tvořit grafy nebo diagramy, a je tedy nutné se opravdu zamyslet nad tím, který z nástrojů budete používat. Používání špatných nástrojů může vést k nesprávnému zobrazení nebo interpretaci předávaných dat a je tedy velice důležité si vybrat kvalitní a spolehlivé nástroje, které jsou schopné zobrazit data korektně a srozumitelně.

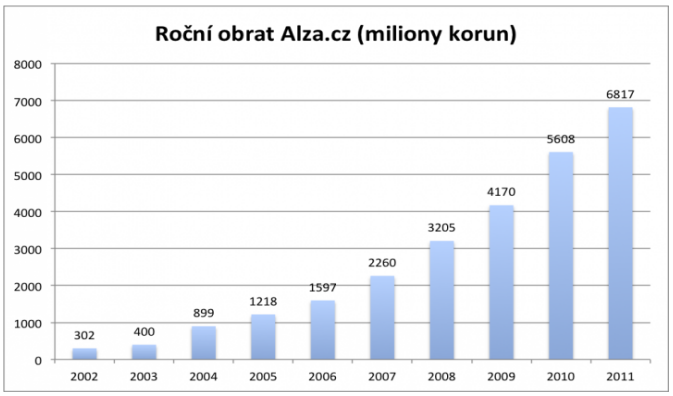
Mezi základní požadavky patří:

1. Funkčnost - Nástroj musí splňovat požadavky uživatele a být schopen zobrazit data tak, jak přesně jsou potřeba.
2. Přesnost - Nástroj musí zobrazovat data správně a musí se s nimi dát pracovat bez chyb.
3. Srozumitelnost - Nástroj by měl zajistit snadné pochopení a interpretaci dat.
4. Použitelnost - Nástroj by měl být snadno použitelný a měl by také být intuitivní a uživatelsky přívětivý.

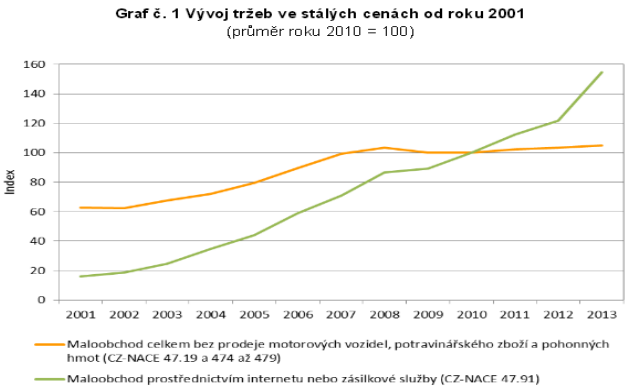
Předchozí krátký výpis požadavků určitě nezahrnuje všechny požadavky, které by měly být kladeny na nástroj, ale jedná se pouze o základní požadavky, které by měl vámi zvolený nástroj vždy splňovat. Mezi dalšími požadavky bychom měli pamatovat i například na to, jestli je daný nástroj jednoduše přenositelný a jestli ho lze jednoduše sdílet. Pokud například chcete výslednou vizualizaci sdílet s ostatními lidmi, může být užitečné najít takový nástroj, který umožňuje snadno exportovat výslednou vizualizaci do různých formátů, jako je PDF nebo obrázek. Následně bychom se také měli zamyslet nad tím, zda je váš nástroj snadno integrovatelný s ostatními nástroji a systémy s nimiž pracujete. Pokud například pracujete s daty, která jsou uložena na cloudovém úložišti, jako je Google Cloud nebo Microsoft Cloud, může být poté vhodné vybrat takový nástroj, který umožňuje jednoduchý přístup k vašim datům a který je schopný je integrovat s dalšími nástroji.

3 Jak vytvořit efektivní vizualizaci

Mezi základními požadavky na efektivní vizualizaci dat hraje klíčovou roli zvolení toho správného grafu nebo diagramu pro vybraná data. Pokud máme například vizualizovat prodeje v letošním roce oproti rokům předchozím, tak k takovému zobrazení se hodí sloupcový, viz obrázek 3.1, nebo spojnicový graf, viz obrázek 3.2.



Obrázek 3.1: Sloupcový graf použitý pro zobrazení vývoje ročních obratů [1]

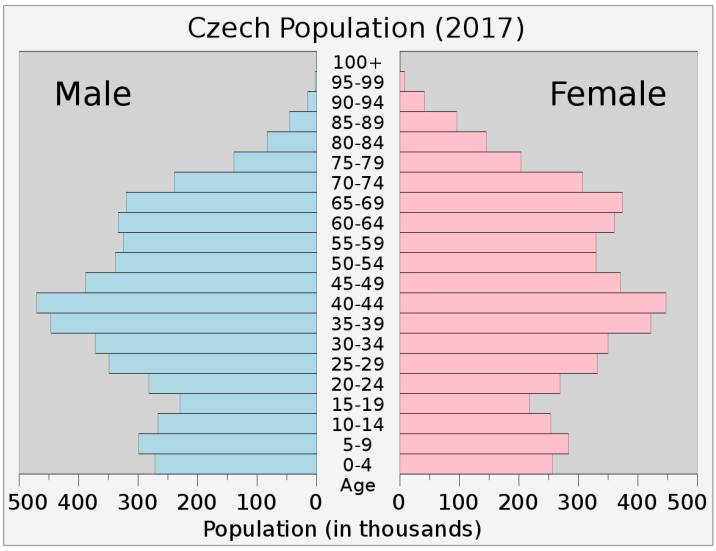


Obrázek 3.2: Spojnicový graf [2]

[1] Zdroj: https://www.lupa.cz/clanky/krasne-nove-grafy-nejvetsich-ceskych-e-shopu-co-se-skryva-za-nimi/

[2] Zdroj: https://www.epravo.cz/_dataPublic/photo/5cbb9b9bc7c76f5c6e051d207f4b735b/SAMAK_graf.png

Pokud ale potřebuje například zobrazit populační pyramidu, tak v tomto případě se sice dá také využít standardní sloupcový graf, ale o mnoho lepší je využít speciální podmnožinu sloupcového grafu, neboli tzv. histogram. Tento typ grafu se primárně používá k vizualizaci rozložení dat v určitém rozsahu. Tyto grafy se obvykle používají pro kvantitativní data, jako jsou četnosti nebo počty. Z tohoto důvodu jsou zcela vhodné pro populační pyramidu, jelikož se na tomto typu grafu dá daleko lépe pozorovat poměr pohlaví mezi věkovými skupinami. Viz obrázek 3.3



Obrázek 3.3: Graf populace ČR v roce 2017 [1]

Výpis základních grafů a k nim jejich využití:

1. Sloupcový graf - Tento typ grafu je dobrý pro porovnávání hodnot v různých kategoriích nebo přes delší časové období. Je dobrý pro zobrazení relativních velikostí a pro porovnávání hodnot v rámci kategorií.

2. Spojnicový graf - Tento graf je obecně dobrý pro zobrazení trendů a vývoje v čase.

3. Mapa - Mapa je dobrá pro zobrazení geografických údajů a pro porovnávání hodnot mezi různými oblastmi.

4. Sankey diagram - Tento typ grafu je dobrý pro zobrazení toku hodnot mezi různými kategoriemi nebo skupinami.

5. Scatter plot - Tento graf je dobrý pro zobrazení vztahu mezi dvěma veličinami a pro identifikaci jejich odchylek.

[1] Zdroj: https://cs.wikipedia.org/wiki/Soubor:Czechpop.svg

Jak je tedy z předchozího krátkého výpisu grafů a jejich využití zřejmé, pro efektivní vizualizaci dat je potřebné mít jasnou představu o tom, jaká data chceme vizualizovat a co chceme pomocí této vizualizace ukázat naší cílové skupině. Pokud známe odpověď na tyto dvě otázky, tak následuje samotný výběr vizualizace. Jak již bylo ukázáno dříve, každý typ vizualizace se hodí na různé typy dat a pro různé účely. Například pro zobrazení výchylek dodávek v rámci několika let se jako první nabízí graf spojnicový nebo sloupcový. Naopak pro tento případ je zcela nevhodný graf výsečového typu nebo graf typu treemap. Tyto typy grafů se totiž spíše využívají pro tzv. kategorická data než pro data numerická.

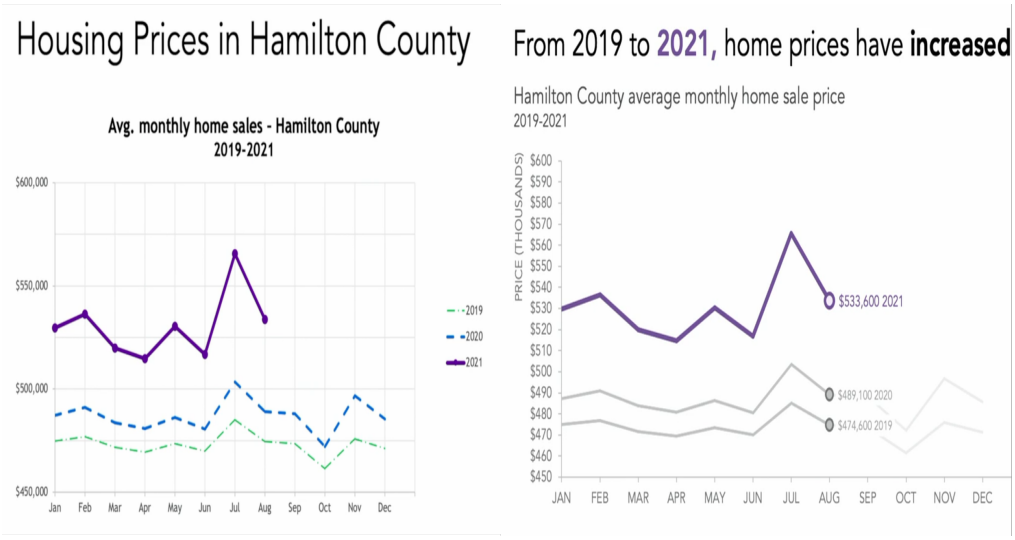
Kategorická data jako taková jsou alternativním druhem dat, která se klasifikují do diskrétních kategorií nebo skupin. Tyto kategorie nebo skupiny nemají mezi sebou žádný vztah a z tohoto důvodu nemají žádné přirozené pořadí. Kategorická data jsou také často vyjadřována pomocí nominální škály, což znamená, že každá kategorie je pouhým označením a nemá přiřazenou žádnou vlastní hodnotu. Pro jednodušší pochopení si můžeme toto představit tak, že kategorická data jsou zpracovávána pomocí četností (počtu výskytů) oproti numerickým. Typickým příkladem kategorických dat jsou například barvy a jak často se vyskytují v sezonní kolekci analyzované společnosti.

Po výběru vhodného typu vizualizace pro naše data práce nekončí a následuje přizpůsobení vybrané vizualizace k jejímu zpřehlednění. Jak napsala C. Knaflic ve své knize [1] "Vizualizace by měla vypovídat o příběhu, který s ní chceme vyprávět." a toho se dá dosáhnout za pomoci již zmíněného přizpůsobení. V následující části budou popsány vybrané možnosti přizpůsobení vizualizace.

3.1 Alterace barev

Jak již bylo částečně ukázáno na obrázku 2.6, výběr vhodné palety barev není radno uspěchat, jelikož špatný výběr může námi vytvořenou vizualizaci ještě více znepřehlednit. Podle webu Chartio [3] by přehledná vizualizace měla obsahovat nanejvýš 10 od sebe odlišných barev, toto pravidlo se řadí mezi tzv. "best practices". Pokud se jedná o jednoduché grafy s jasně definovanými kategoriemi, je doporučováno využití maximálně 5 barev. Ve vizualizaci s více barvami než je tzv. "best practice" totiž dochází k problému se

splýváním barev, a tudíž je těžké od sebe odlišit jednotlivé kategorie. Mezi další špatné návyky patří tzv. "protáčení barev", kde po využití celé naší palety barev pro část našich kategorií znovu paletu "protočíme" a začneme barvy znovu využívat pro zcela odlišné kategorie. Tím ztratí jednotlivé kategorie svojí unikátnost, což vede k dalšímu znepřehlednění dané vizualizace. V tomto případě je více vhodné zanedbatelné kategorie sloučit do jedné, kterou označíme jako ostatní. Naopak pokud chceme například ukázat hustotu v jedné kategorii, tak místo několika barev je vhodné využití odstínů jednoho barevného spektra. Tím zajistíme jednoduchou možnost detekovat, kde je daná kategorie silnější a kde je tomu naopak.



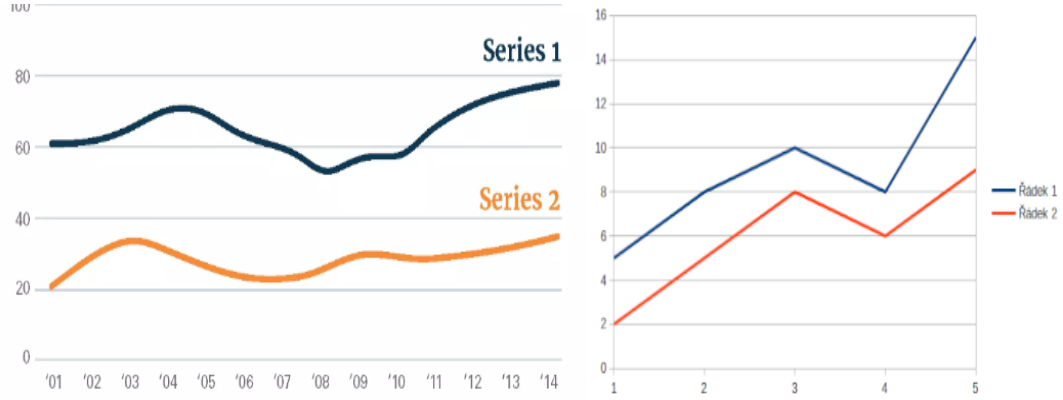
Obrázek 3.4: Porovnání tzv. point of focus [1]

Na obrázku 3.4 lze vidět přímý příklad, jak se za pomocí přizpůsobení dá pozorovatelova pozornost neboli tzv. "point of focus" zaměřit na určitou oblast. V levém obrázku před úpravou pozorovatel z velké části neví, na co se přesně zaměřit, jelikož všechny informace vypadají jako důležité. To je způsobeno tím, že všechny informace jsou vedeny jinými barvami a s různým typem linie. Je zde sice připojena i legenda, ale nejpodstatnější údaj je zde vložen až jako poslední. Tomuto problému se pravá strana obrázku vyvarovala tak, že pro všechny informace, které jsou důležité, zvolila sytou barvu a ostatní méně podstatné informace potlačila méně sytou barvou (v našem případě šedou). Tímto se pozorovatelova pozornost přesunula na informace důležité pro daný graf, čímž se graf stal čitelnějším a přehlednějším.

[1] Zdroj: https://community.storytellingwithdata.com/videos/emphasize-an-insight

3.2 Popisky grafické vizualizace

Důležitou věcí při přizpůsobovaní grafu je také vhodná volba popisků. Většina datových řad obsahuje mnoho různých atributů pro daná data, ale pokud chceme vyprávět příběh, tak je potřeba z těchto atributů vybrat ty podstatné, které nám pomohou nasměrovat příběh požadovaným směrem. Pokud chceme například hovořit o růstu prodejů mobilních telefonů mezi různými modely dané společnosti Z, tak mezi klíčové atributy kromě jména prodaných modelů bude nejspíše patřit i počet prodaných kusů a v jakém časovém období se tyto kusy prodaly (například v jaký měsíc). Oproti tomu jaké barvy a s jakou verzí OS tyto mobilní telefony byly vybaveny už v rámci vyprávěného příběhu ztrácí smysl zmiňovat. V tomto případě se tedy jako popisky grafu nabízí zvolit počet prodaných kusů, název modelu a název měsíce, ve kterém se tento model prodal. Jelikož ale při 2D vizualizaci máme pouze dvě osy X a Y, tak pro 3 klíčové popisky je nutno jeden z nich zakomponovat jiným způsobem než jako popisek hlavní osy X a Y. Toho se dá docílit například vložením dodatečného popisku do datové oblasti.



Obrázek 3.5: Porovnání legend [1]

Popisky v datové oblasti jsou jedním ze způsobů, jak pozorovateli nebo cílové skupině přenést dodatečné informace, které by nebyly například tak přehledné, pokud by byly obsažené jako dodatečná informace pro osu X nebo Y. V praxi často tvoří tzv. legenda, která slouží k vysvětlení, co jednotlivé hodnoty (pro nás například sloupce nebo linie) znamenají [4]. Bohužel většinou jsou tyto legendy daleko od samotného grafu a působí tedy spíše jako samostatná komponenta než jako integrovaná komponenta do celku. Nejčastější příklad tohoto chování je vidět na pravé straně obrázku 3.5, kde je legenda na pravé straně daného grafu, takže pozorující jsou nuceni vždy

[1] Zdroj: vlastní + https://policyviz.com/2018/08/07/dataviz-cheatsheet/

se podívat na graf a následně na legendu, aby byli schopni rozlišit, co jednotlivé linie znamenají. Pokud by navíc legenda byla obsáhlejší, tak se její problém ještě prohlubuje, jelikož se stane velmi nepřehlednou. V takovýchto případech je lepší legendu přesunout hned pod nadpis daného grafu a nebo využít tzv. vkládání legendy přímo do datové vizuálu. Vkládání legendy do datové sady je velmi účinné přizpůsobení, jejíž pomocí jsme schopni přehledně doplnit požadované informace do grafu, a to takovým způsobem, že graf zůstane stále přehledný. Navíc je zde i výhoda v tom, že tak nevynucujeme přesouvání pozornosti pozorovatele mezi různými komponentami a tím je i snížené riziko, že daný pozorovatel se přehlédne nebo něco opomene. Příklad integrované legendy v grafu lze vidět na levé straně obrázku 3.5.

3.2.1 Markery

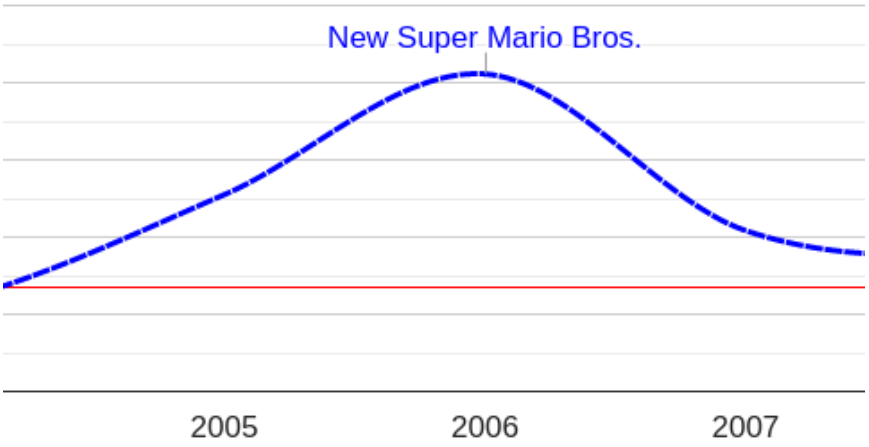
Mezi další možnosti jak přenášet dodatečné informace pozorovatelům prostřednictvím popisků v grafických vizualizacích jsou markery. Markery jsou vizuální prvky, které se přidávají ke grafickým objektům a slouží k označení nebo značení určitých vlastností či informací. Markery mohou být různých typů a mají za cíl poskytnout užitečné doplňující informace, které pomáhají lépe porozumět datům a výsledkům vizualizace.

Existuje několik způsobů, jak lze markery využít k přenosu dodatečných informací. Jedním z nejčastěji používaných způsobů je přidání popisků s názvy nebo popisy grafických objektů. Tímto způsobem lze identifikovat jednotlivé prvky ve vizualizaci a zdůraznit důležité milníky či klíčové body. Například v grafu by markery mohly označovat významné hodnoty na grafických sloupcích nebo konkrétní body na časové ose viz obrázek 3.6.

Dalším způsobem využití markeru je jejich interaktivní charakter. Uživatelé mohou po umístění kurzoru myši na marker zobrazit podrobnější informace přímo v rámci vizualizace. To může být realizováno úpravou samotného grafu, kdy se zvýrazní spojení nebo hodnoty související s daným markerem. Alternativně se může objevit tzv. tooltip, což je malé vyskakovací okno, které poskytuje dodatečné vysvětlení nebo kontextové informace k markru. Tento přístup umožňuje poskytnout rozšířené informace a podrobnosti, aniž by zatížil čitelnost samotné vizualizace.

Využití markrů v grafických vizualizacích přináší přidanou hodnotu tím, že umožňuje efektivnější a bohatší prezentaci dat. Skrze tyto popisky mohou

uživatelé lépe porozumět různým aspektům vizualizace a provádět detailnější analýzu dat. Interaktivní povaha markrů navíc umožňuje uživatelům rychlý přístup k dalším informacím bez narušení celkového kontextu vizualizace.



Obrázek 3.6: Příklad markeru a baseline na časové ose (výřez)

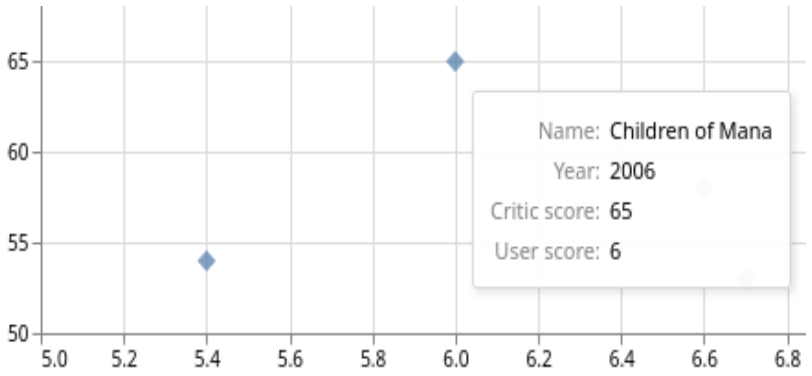
3.2.2 Tooltip

Mezi další zajímavé nástroje pro přenášení dodatečných informací pomocí popisků v grafických vizualizacích je již zmíněný tooltip. Jak již bylo zmíněno výše, tento interaktivní prvek se aktivuje při umístění kurzoru myši na specifický prvek, jako je bod nebo oblast s daty na grafu. Tento prvek se zobrazuje jako malé vyskakovací okno s relevantními informacemi vzhledem k vybranému specifickému prvku.

Tooltipy v rámci kontextu vizualizací dat přináší hned několik výhod. První výhodou je možnost poskytnout uživatelům detailní informace o právě zkoumaných prvcích. Například na obrázku 3.7 zobrazující hodnocení uživatelů a kritiků k jednotlivým vydaným hrám od společnosti Nintendo, tooltip zahrnuje nejen samotné hodnocení kritiků a uživatelů, ale také další relevantní atributy, jako je název produktu (atribut Name) a rok, ve kterém hodnocená hra byla vydána (atribut Year).

V množině výhod tooltipu je také schopnost zachovat přehlednost celé vizualizace. Díky své kompaktní povaze a omezenému zobrazování na malé vyskakovací okno nezabírají tooltipy příliš místa na obrazovce. To umožňuje

uživatelům snadno získat potřebné informace, aniž by se narušila celková čitelnost vizualizace. Lze tedy říci, že tooltipy umožňují uživatelům provádět efektivnější a bohatší analýzu dat a interaktivita, kterou poskytují, přispívá k lepšímu porozumění vizualizovaným datům.



Obrázek 3.7: Příklad tooltipu pro hodnocení Nintendo her

3.2.3 Baseline

Dalším významným prvkem popisků grafických vizualizací je baseline, která slouží jako referenční hodnota pro porovnání s ostatními datovými prvky. Baseline umožňuje uživatelům rychlé a intuitivní porovnání hodnot a identifikaci vztahů mezi daty. Typickým zobrazením baseline je horizontální čára, jak je ukázáno na obrázku 3.6. Nicméně, baseline není omezena pouze na horizontální čáru a může mít různé formy, včetně přímek, křivek nebo jiných grafických prvků.

Primárním cílem baseline ve vizualizacích je umožnění rychlého rozpoznání trendů, anomálií nebo odchylek v datech. Porovnání dat s baseline také pomáhá uživatelům získat komplexní kontext a lépe porozumět datům a jejich významu. Sekundárním přínosem baseline je, že slouží jako vizuální orientační bod. Uživatelé tedy mohou snadno odhadnout, zda je hodnota prvku nad nebo pod baseline, což umožňuje uživateli si vytvořit představu o tom, jaký "příběh" daná vizualizace vypráví. Baseline lze využít v různých typech grafických vizualizací, jako jsou spojnicové grafy, sloupcové grafy, scatter ploty nebo další. Její umístění a forma závisí na specifických potřebách a charakteru dat. V některých případech může být umístěna jako horizontální čára přímo na ose grafu, čímž poskytuje pevný bod pro srovnání hodnot. V jiných případech může být baseline vyjádřena jako relativní

hodnota vůči jiným prvkům, například jako průměrná hodnota datové sady nebo jako referenční hodnota získaná z jiného zdroje.

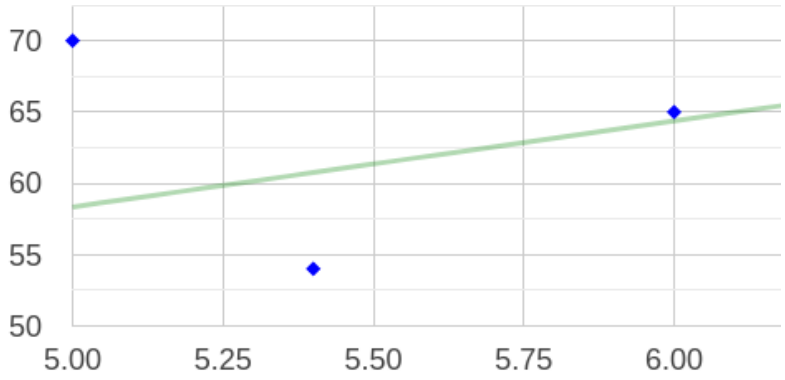
3.2.4 Gridlines

Gridlines jsou dalším důležitým prvkem grafických vizualizací, který přispívá k přehlednosti a interpretaci dat. Jedná se o sítě rovnoběžných horizontálních a vertikálních čar, které se překrývají na grafu a tvoří povětšinou mřížku viz obrázek 3.8. Gridlines slouží k vizuálnímu zarovnání datových prvků a umožňují uživatelům snadno odhadnout hodnoty na osách grafu. Implementace gridlines do výsledné vizualizace má hned několik výhod. Jako hlavní výhodu lze pokládat její přispění k lepší orientaci a porozumění vizualizovaným datům. Gridlines umožňují uživatelům snadno sledovat hodnoty na ose a provádět rychlé porovnání mezi různými datovými body. Díky zarovnanému vzhledu, který gridlines poskytují, je možné snadno identifikovat vzorce, trendové linie a odchylky ve vizualizaci.

Důležitou funkcí gridlines je také jejich schopnost vizuálně oddělit jednotlivé oblasti na grafu. Vertikální gridlines rozdělují prostor na horizontální osách a umožňují uživatelům snadno identifikovat, kde se nacházejí jednotlivé hodnoty nebo kategorie. Horizontální gridlines pak rozdělují prostor na vertikálních osách a umožňují rychlé určení hodnot na základě jejich výšky nebo rozpětí. Tato vizuální separace a organizace přispívá k lepší orientaci a porozumění datům na grafu.

Gridlines také přidávají do vizualizace estetický prvek. Kombinace rovných a pravidelně rozložených čar gridlines vytváří vizuálně příjemný a vyvážený vzhled. To je důležité z hlediska přitažlivosti vizualizace a její schopnosti upoutat pozornost uživatele. Zarovnané gridlines přispívají také k celkové čitelnosti grafu a usnadňují rychlé vnímání a interpretaci dat.

Při implementaci gridlines je však třeba zohlednit několik faktorů, jako je vhodná hustota čar, tloušťka, barva a styl. Příliš husté gridlines by mohly způsobit přeplnění grafu a ztížit čtení hodnot, zatímco příliš řídké gridlines by mohly ztratit svůj účel v poskytování vizuálního zarovnání. Z toho vyplývá, že volba správného stylu a vzhledu gridlines je velmi důležitá pro dosažení optimálního výsledku a přizpůsobení se specifickým potřebám vizualizace.



Obrázek 3.8: Příklad gridlines + trendline (výřez)

3.2.5 Filtrace dat

Velmi zajímavým přizpůsobením vizualizace je i tzv. filtrace dat, která umožňuje uživatelům interaktivně omezit množinu dat zobrazenou na grafu. Jedná se o nástavbu běžné filtrace, která se provádí v rámci předzpracování dat. Tato filtrace dat funguje velice podobně jako její tradiční část v rámci předzpracování jen s tím rozdílem, že je prováděna nad daty, která jsou již vhodná a správně naformátovaná pro danou vizualizaci. Samotná filtrace se dá přirovnat k jakémusi užšímu výběru z již zcela validních dat podle aktuálních potřeb nebo zájmů dynamicky upravovat data, která jsou již zobrazena ve vizualizaci. To znamená, že uživatelé mohou interaktivně měnit parametry filtrace a okamžitě sledovat, jak se tato změna promítá do vizualizace. To uživateli poskytuje možnost zkoumat různé aspekty dat a odhalovat dříve přehlédnuté vzorce, trendy nebo vztahy. Pro příklad si můžeme uvést marketingové oddělení fiktivní firmy XYZ, kde hlavní analytik dostane automaticky vytvořený graf prodejů za posledních 30 let. Úkolem tohoto analytika je provést analýzu poslední 10 let a upravit graf tak, aby byly zřetelné všechny důležité milníky a klíčové body. Toho analytik dosáhl tak, že graf omezil za pomocí filtrace na posledních 10 let a označil jednotlivé důležité milníky a klíčové body do grafu. Následně za pomocí další filtrace si zobrazil jednotlivé roky a jejich milníky a takto vytvořené grafy prezentoval vedení, kde jednotlivé významné body byly vidět jak v rámci jednotlivých let, tak také v kontextu posledních 10 let společně.

4 Možnosti vizualizace

4.1 Množina funkcionalit

Na základě kapitoly o efektivní vizualizaci a knihy od C. Knaflic [1] byla vypracována sada požadovaných funkcionalit, kterou by ideální knihovna měla obsahovat. Následuje detailní rozpis jednotlivých funkcionalit:

1. Interaktivita grafů - umožnění přidání interaktivních prvků do vizualizace. Jedná se především o možnost zobrazení dodatečných informací po vybrání kurzorem (za pomocí tooltipu nebo bez), zvýrazňování dat za pomoci změny vizuálu jednotlivých prvků jako je tloušťka ohraničení, barevnost výplně a dalších interakcí, které umožní uživatelům prozkoumat data a získat podrobnější informace.

2. Definování vlastní barvy - definování vlastní barevné palety pro vizualizaci, kterou je možno přizpůsobit specifickým potřebám. Definování vlastní barvy zahrnuje všechny obecně známé možné druhy kódování barev jako je HEX, RGB/RGBA nebo CMYK.

3. Škálovatelnost os - umožnění různých druhů škálování os grafu, aby se zajišťovala dobrá čitelnost a vhodné zobrazení dat při různých rozsazích. To zahrnuje úpravu druhů škálování os jako je logaritmické nebo lineární škálování. Do této funkcionality patří také automatické nastavení os bez nutnosti přímého zásahu do konfigurace vizualizace (když všechna data budou 50+, tak nemá povětšinou smysl generovat osu od 0).

4. Úprava typů čar v grafu - měnění typů čar v rámci vizualizace zcela nezávisle na sobě. Do těchto typů čar primárně patří plná a přerušovaná čára.

5. Nastavení baseline - přidání a nastavení vizuálu baseline pro vizualizaci. V rámci tohoto přidání by měla umožňovat přidat i popisek k jednotlivé baseline (například číslo nebo krátký text).

6. Slicing - vykreslení okna pro vizualizaci pouze v definovaném rozsahu (většinou je pevně definované jako třeba rok 2000 - 2010), ale ostatní data jsou stále v rámci celé vizualizace uložená, pouze nejdou vidět (jsou mimo záběr okna).

7. Dodatečné popisky - přidaní dodatečných popisků k bodům, liniím nebo oblastem vizualizace, což usnadňuje identifikaci a interpretaci konkrétních dat. Popisky by měly jít přidat v rámci celé vizualizace, což zahrnuje i oblast mimo datové body a linie.

8. Modifikace dat - podpora různých úprav a stylizací dat. Jedná se především o změnu velikosti, barvy a tvaru datových bodů, aby bylo možné dosáhnout chtěného vizuálního efektu.

9. Posunutí báze - posunutí báze (čímž se především rozumí posunutí os) vizualizace na uživatelem předem definovanou hodnotu. Tato funkcionalita umožňuje lepší vizualizaci dat a umožňuje zobrazit data v jiném kontextu nebo s jiným přesahem.

10. Filtrace dat - schopnost využití dodatečné filtrace dat. Jak již bylo řečeno v kapitole 3.2.5, tato funkcionalita by měla uživatelům umožnit zobrazit pouze určitý rozsah dat v rámci vizualizace, kde se neodpovídající data z vizualizace odeberou (což je rozdíl oproti funkcionalitě slicing).

11. Stylizování os grafu - úprava vzhledu os, jako je změna velikosti i fontu písma, formátování čísel, přidání jednotek k jednotlivým osám, a to zcela nezávazně na sobě. Obecně vzato úkolem této funkcionality je přispívání k estetickému vzhledu a efektivnímu sdělení informací v rámci vizualizace.

12. Popisky os - přidání popisků k osám grafu, aby bylo jasně vyjádřeno, jaká data osy představují. To zahrnuje i sekundární popisky v rámci jedné osy, které slouží k předání dodatečných informací k primárnímu popisku.

13. Výplň datové sady - schopnost vyplnění množiny datového sady. Využití výplně datové sady je zejména důležité při vizualizacích, které se zaměřují na zobrazení plochy, kterou sada dat pokrývá.

14. Markery - přidání v rámci vizualizace markerů k určitým bodům grafu. Tato funkcionalita usnadňuje identifikaci klíčových hodnot nebo událostí, jak bylo již zmíněno v rámci kapitoly 3.2.1.

15. Modifikace legendy - úprava vzhledu legendy grafu, jako je změna velikosti písma, uspořádání položek a přidání popisků, což zvyšuje srozumitelnost a přehlednost grafu.

16. Předdefinované sady barev - předdefinované sady barev v rámci knihovny, které uživatelé mohou použít při vizualizaci dat. Tyto sady totiž usnadní výběr harmonických a atraktivních barevných schémat vizualizace.

4.2 Množina vizualizací

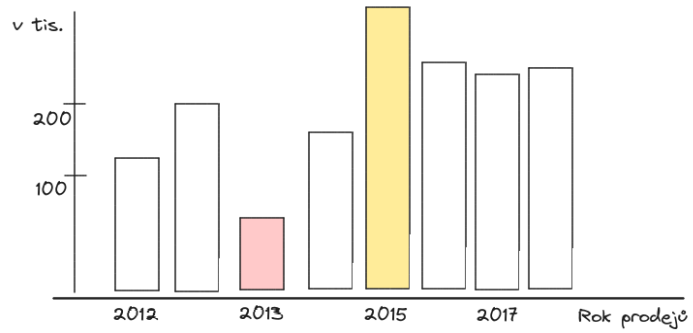
Za pomocí poznatků získaných z webového článku Oracle [5], Datatoviz [6] a knihy od C. Knaflic [1] byla vybrána také množina grafů, na kterých byly tyto funkcionality vyzkoušeny. Ke každému grafu byla na základně vhodnosti přidělena podmnožina funkcionalit. Jedná se především o grafy, které jsou globálně hojně využívané a do kterých ale nespadají sporné typy grafů jako například pie chart. Podle velké části odborníků totiž není tento typ vizualizací zcela vhodný, jak již bylo zmíněno v kapitole 2.1. Následuje detailní popis jednotlivých druhů vizualizací společně s funkcionalitami a k nim připravených obecných příkladů vizualizací.

4.2.1 Bar chart

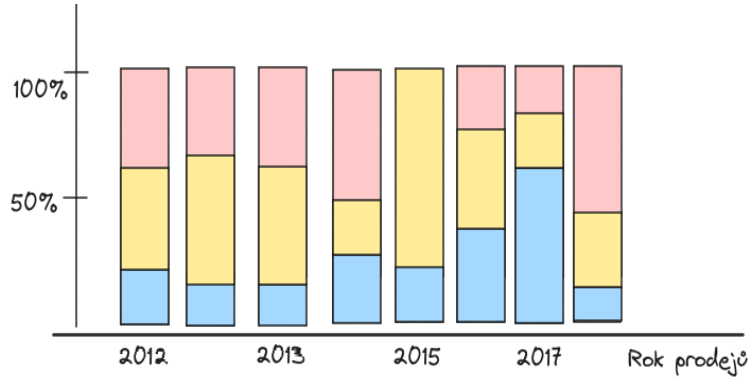
Bar chart neboli sloupcový graf je velmi používaný vizuální nástroj především pro prezentaci kvantitativních a kategorických dat. Jedná se o graf, který je tvořen souborem svislých sloupců, kde každý sloupec reprezentuje jednu kategorii nebo skupinu dat. Výška každého sloupce odpovídá hodnotě, kterou reprezentuje. To umožňuje snadné vizuální porovnání mezi různými skupinami. K tomuto grafu byly přiděleny tyto funkcionality:

1. Interaktivita grafů
2. Definování vlastní barvy
3. Škálovatelnost os

V rámci tohoto grafu byl také definován požadavek na samostatné zobrazení za pomocí kvantitativních a kategorických dat. V případě kategorických dat byla dále množina funkcionalit rozšířena o položku 100% skládaného sloupcového grafu, jelikož se jedná o velmi výhodnou vizualizaci dat v rámci proporcionálního zobrazení. Všechny sloupce jsou totiž stejně vysoké (100%) a jejich složení odpovídá proporčním hodnotám jednotlivých skupin.



Obrázek 4.1: Návrh sloupcového grafu s požadovanými funkcemi

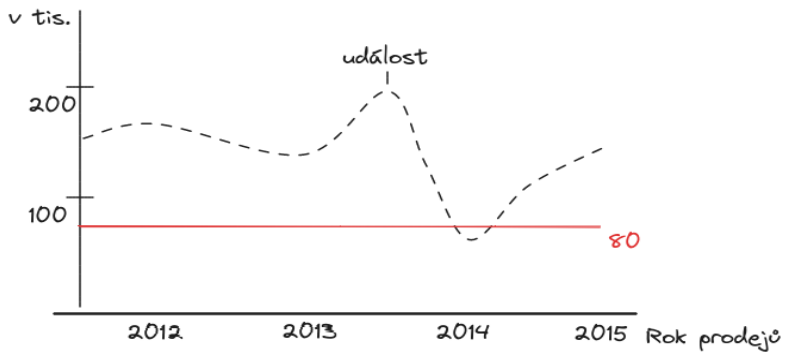


Obrázek 4.2: Návrh 100% sloupcového grafu s požadovanými funkcemi

4.2.2 Line chart

Line chart neboli spojnicový graf je další z velmi používaných nástrojů pro prezentaci kvantitativních dat. Jedná se o graf, který je tvořen za pomocí jedné nebo více linií. Každá z již zmiňovaných linií představuje jeden tzv. data set, který za pomocí prokládání bodů vykresluje požadované linie. K tomuto grafu byly přiděleny tyto funkcionality:

1. Úprava typů čar v grafu
2. Nastavení baseline
3. Slicing
4. Dodatečné popisky

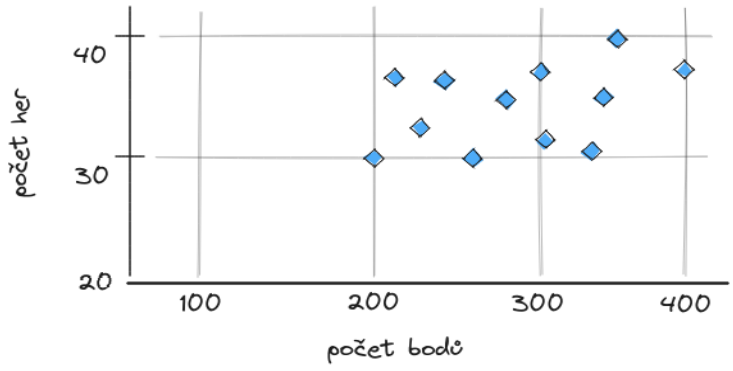


Obrázek 4.3: Návrh spojnicového grafu s požadovanými funkcemi

4.2.3 Scatter plot

Scatter plot neboli bodový graf je velmi populární graf v rámci vizualizace vztahu mezi dvěma proměnnými. Hlavní úlohou tohoto typu grafu je ukázat rozložení bodů na souřadnicové síti a odhalit případné vzorce, shluky nebo vztahy mezi proměnnými. Z velké části se tento graf využívá v oblastech jako je statistika, data science, ekonomie, medicína a další. K tomuto grafu byla přiřazena tato množina funkcionalit:

1. Modifikace dat
2. Posunutí báze
3. Filtrace dat

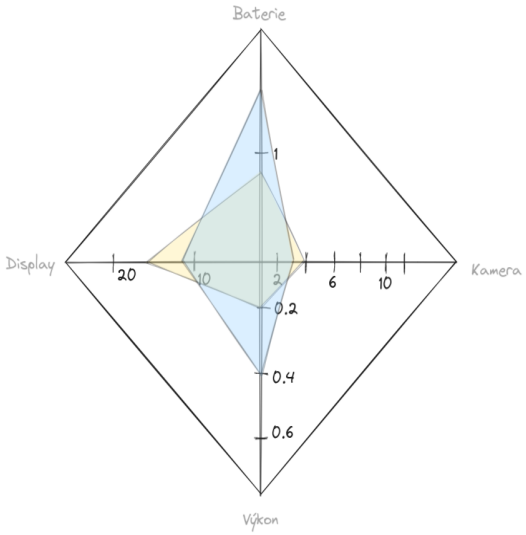


Obrázek 4.4: Návrh bodového grafu s požadovanými funkcemi

4.2.4 Radar chart

Radar chart, spider chart nebo paprskový graf je pokročilý graf v rámci obecných vizualizací. I přes jeho komplexnost je velmi využívaný v oblasti porovnávání (obzvláště v technice), kde v rámci několika kategoriích se přidělují body a následná vizualizace jednoznačně vypovídá o plusech a mínusech daného produktu napříč kategoriemi. Jako příklad si můžeme uvést telefon od společnosti X, kde jsou kategorie výdrž, výkon, kamera a display. Celková plocha daného produktu nám v rámci porovnání ihned řekne, jestli si tento model společnosti X vede lépe než jiný s kterým ho porovnáváme. Toto je jen jeden z mnoha možných příkladů použití tohoto grafu. K tomuto grafu byly přiděleny tyto funkcionality:

1. Stylizování os grafu
2. Popisky os
3. Výplň datové sady

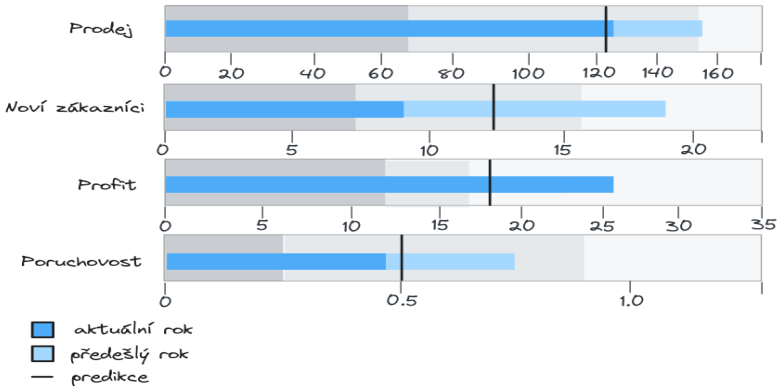


Obrázek 4.5: Návrh paprskového grafu s požadovanými funkcemi

4.2.5 Bullet chart

Bullet chart je další z řady populárních pokročilých grafů v rámci obecných vizualizací. Jedná se o kompaktní a informativní graf, který je často využíván pro porovnávání výkonnosti a cílů. Je navržen tak, aby zobrazoval několik měřítek nebo ukazatelů současně, což umožňuje uživatelům rychle porovnat skutečné hodnoty s cílovými hodnotami nebo s jinými referenčními body. Bullet chart tedy oproti sloupcovému grafu slouží spíše k porovnání skutečných hodnot s definovanými cíli v rámci jednoho kompaktního grafu. K tomuto grafu byla přidělena tato množina funkcionalit:

1. Stylizování os grafu
2. Popisky os
3. Výplň datové sady



Obrázek 4.6: Návrh bullet grafu s požadovanými funkcemi

4.3 Data

Pro vizualizaci byly vybrány 2 datové sady, jelikož obsahují zajímavé poznatky z herního průmyslu. První sadou dat je sada obsahující herní prodeje od roku 1980, které přesáhly 100 000 jednotek prodejů. Tato datová sada byla získána z webu kaggle, který slouží jako globální platforma pro datové sady z celého světa [7]. Z tohoto velkého data setu byly extrahovány za pomocí Pythonu a knihovny Pandas globální prodeje jednotlivých her společně s vydavatelem a rokem vydání. Následně byla nově vytvořená datová sada dodatečně transformována, aby jednotlivé hry od stejného vydavatele a se

stejným rokem vydání byly sdruženy do jednoho zápisu pro daný rok. V poslední úpravě těchto dat byla datová sada očištěna od všech společností, které nevydávaly větší množství herních produktů v rámci sledovaného období.

Stejně jako první datová sada, tak i druhá datová sada se zaměřovala na herní průmysl, jelikož obsahuje poměrně zajímavé poznatky. Přesněji se jedná taktéž o prodeje herního průmyslu do roku 2016. Z této datové sady byla extrahována herní data pouze od společnosti Nintendo. V rámci další transformace datové sady od Nintenda byla tato datová sada očištěna pouze na jména jednotlivých her, rok vydání, user score a critic score, kde rok vydání byl omezen pouze na sledované roky 2005 a 2006. Originální datová sada byla rovněž získána z webu kaggle [8]. Takto vytvořené datové sady budou v rámci následujících částí vizualizovány za pomocí javascriptových knihoven a grafů, které byly definovány v předchozí kapitole. První datová sada je v rámci kategorie dat zařaditelná spíše k datům kategorickým, jelikož obsahuje hned několik odlišných společností, které jsou na sebe z velké části nezávislé. Oproti tomu druhá datová sada má ze své podstaty spíše blíže k datům numerickým.

V kontextu následné vizualizace je úkolem první sady dat vykreslit poměry prodejů jednotlivých společností napříč roky za pomocí skládaného sloupcového grafu a dalších, které jsou k těmto datům vhodné. V rámci druhé datové sady se jedná především o vizualizaci poměrů jednotlivých hodnocení uživatelů a kritiků v návaznosti na to, jak si společnost Nintendo vede v uspokojení jejich zákazníků a kritiků. Originální datové sady lze nalézt v přílohách v podadresáři Data/původní. Modifikované datové sady lze nalézt v adresáři Data. Ve složce Data/Scripts je také možné vidět skripty pro vytvoření vybraných modifikací za pomocí knihovny Pandas v Pythonu.

V případě datových sad uvedených v této práci jsou veškeré informace uloženy ve formátu CSV. To znamená, že jednotlivé hodnoty prodejů her, vydavatelů, roků vydání a dalších atributů jsou uloženy v textovém souboru, kde jsou odděleny čárkami. Tento formát umožňuje snadné načítání a zpracování dat pomocí různých programovacích jazyků nebo nástrojů, včetně vizualizačních knihoven v JavaScriptu.

4.4 JavaScriptové Knihovny

Tato kapitola definuje, co je to JavaScript, jaké využívá možnosti pro vykreslení vizualizace a pojednává o množině vybraných JavaScriptových knihoven. Knihovny byly vybírány především na základě jejich popularity, četnosti použití a zajímavosti.

4.4.1 JavaScript

Na základě informací z dokumentace [9] a W3 webu [10] je JavaScipt (JS) populární programovací jazyk používaný pro vývoj webových stránek a interaktivních webových aplikací. Byl vyvinut firmou Netscape v roce 1995 a je základem pro další jazyky jako je například TypeScript. JavaScript je skriptovací jazyk, což znamená, že je interpretován přímo v rámci webového prohlížeče a nepotřebuje kompilaci. JavaScript je nezávislý na platformě, takže může běžet na všech známých operačních systémech. Je podporován také ve většině moderních webových prohlížečů jako je Mozilla Firefox, Google Chrome, Microsoft Edge nebo Safari.

Hlavní využití JavaScriptu spočívá v manipulaci s obsahem webových stránek, interakci s uživateli, validaci formulářů, animacích, asynchronním načítání dat a komunikaci se serverem pomocí asynchronních požadavků AJAX (Asynchronous JavaScript and XML). Tím umožňuje načítání dat z webového serveru poté, co je stránka načtena, posílání dat serveru na pozadí a aktualizaci stránky bez nutnosti jejího znovunačítání. JavaScript je také využíván pro vývoj webových frameworků a knihoven, jako je například React, Angular nebo Vue.js, které usnadňují vývoj komplexních webových aplikací.

Existují dvě hlavní technologie, které JavaScript využívá pro vykreslování vizualizací: SVG (Scalable Vector Graphics) a HTML5 Canvas.

SVG je značkovací jazyk, který umožňuje popis vektorové grafiky pomocí XML syntaxe. JavaScript může generovat a manipulovat s SVG elementy a atributy, což umožňuje vytváření a úpravy různých grafických prvků. SVG je vhodné pro statické a interaktivní vizualizace, zejména pokud se zaměřujete na vektorovou grafiku a manipulaci s prvky pomocí CSS a JavaScriptu.

HTML5 Canvas na druhou stranu poskytuje programovatelné prostředí pro vykreslování 2D grafiky pomocí JavaScriptu. Místo vytváření a manipu-

lace s jednotlivými grafickými objekty jako v případě SVG, Canvas umožňuje přímo vykreslovat pixely na plátno. To umožňuje větší kontrolu nad jednotlivými pixely a efektivní vykreslování složitých vizualizací, ale vyžaduje také více práce při manipulaci s grafickými objekty.

4.4.2 Metrics Graphics

Dle oficiálního textu na webu knihovny Metrics Graphics [11] je tato knihovna postavena jako nadstavba více komplexní knihovny D3 a je optimalizována na efektivní vizualizování a sestavování především dat časových řad. Je schopná vytvářet běžné typy grafiky konzistentním a citlivým způsobem.

Tato knihovna opravdu poskytuje poměrně jednoduchou nadstavbu na mnohem komplexnější knihovnu D3 a zároveň umožňuje poměrně jednoduchou cestu k vizualizaci časových řad. Knihovna ke své vizualizaci využívá Scalable Vector Graphics(SVG), jelikož přímo nevyužívá HTML5. Kromě časových řad tato knihovna také poskytuje možnosti vizualizace za pomocí sloupcových grafů a scatter plotu. Tyto funkcionality se ale v tuto chvíli nachází v rámci tzv. experimentální fáze. To znamená, že ne vše funguje přesně tak, jak by mělo, a některé funkcionality teprve budou implementovány. I přes tzv. experimentální fázi této knihovny se jedná o knihovnu, která má čas své největší slávy prozatím za sebou. V tuto chvíli sice probíhá částečná restrukturalizace celé knihovny a její "reinkarnace", ale to způsobilo, že dokumentace k aktuální verzi knihovny neexistuje a uživatel je tedy nucen využívat starší verze a k nim dohledávat dokumentaci za pomocí nějakého "time machine" webu.

Pokud se vám ale povede nalézt starší knihovnu a společně s ní i její dokumentaci, tak se jedná o poměrně vzhledný a stabilní způsob, jak vizualizovat vaše požadovaná data (především v rámci časových řad).

4.4.3 Google Charts

Google Charts je vysoce populární JavaScriptová knihovna pro vizualizaci dat na webových stránkách. V oficiální dokumentaci se prezentuje jako nástroj, který poskytuje perfektní možnosti vizualizace dat přímo na vaší stránce [12]. Za pomoci této knihovny je možné snadno vytvářet různé typy grafů, včetně sloupcových, liniových, histogramů, scatter plotů a mnoho dalších. Jednou ze značných výhod této knihovny je právě její široká popularita a podpora. Jedná se totiž o knihovnu, která byla vytvořena a je stále udržo-

vána společností Google. To přináší především jistotu spolehlivost, dobré dokumentace a dlouhodobého vývoje. To znamená, že se knihovna pravidelně aktualizuje a vylepšuje, což umožňuje využívat nejnovějších technologií a funkcí.

Další výhodou této knihovny je i její snadná implementace a použití v rámci kódu. Knihovna totiž poskytuje přehledné a intuitivní rozhraní, které umožňuje rychlé a efektivní vytváření požadovaných vizualizací. Navíc za pomoci velmi podrobné a přehledné dokumentace je snadné pochopit a využít různé funkce a možnosti knihovny. V rámci této výhody stojí za zmínku i dedikované fórum, kde je možné najít pomoc v rámci celé komunity Google Charts.

Díky svojí popularitě a širokému rozsahu funkcí se tato knihovna stala velice oblíbenou volbou mnoha vývojářů a designérů. Její schopnost vizualizovat data v atraktivní a interaktivní formě pomáhá předat informace způsobem, který je snadno srozumitelný. Google Charts tyto atraktivní vizualizace vytváří za pomocí SVG jako většina běžných knihoven. Navíc tato knihovna poskytuje i možnost využití jejího API, což umožňuje používání této knihovny při komplexnějších operací.

Vzhledem k výše zmíněným výhodám a popularitě je Google Charts jedním z hlavních hráčů v oblasti vizualizace dat pomocí JavaScriptu a poskytuje spolehlivé a efektivní řešení pro tvorbu poutavých a informativních vizualizací na webových stránkách.

4.4.4 Vega

Vega je mocná JavaScriptová knihovna pro deklarativní vizualizaci dat, jak se lze dočíst z oficiálního webu knihovny [13]. Tato knihovna je navržena takovým způsobem, aby umožňovala tvorbu interaktivních a složitých vizualizací pomocí deklarativního jazyka popisu vizuálních prvků.

Vega se především zaměřuje na oddělení dat od vizualizace a poskytuje tak uživatelům možnost popsat vizualizaci pomocí strukturovaného JSON formátu. Tímto způsobem lze jednoduše definovat různé vizuální prvky, jako jsou grafy, mapy, časové osy a další, a specifikovat jejich vlastnosti a chování. Dalším samostatným modulem k Vega je Vega-lite, která poskytuje ještě vyšší úroveň abstrakce a zjednodušuje proces tvorby vizualizací. Vega-lite totiž umožňuje uživatelům popsat vizualizaci pomocí jednoduchých dekla-

rativních syntaxí a automaticky generuje kompletní specifikaci Vega.

Jednou z klíčových vlastností Vegy je také její flexibilita a rozšiřitelnost. Knihovna poskytuje robustní JavaScriptové API, které umožňuje uživatelům vytvářet vlastní vizuální prvky, transformace dat a interakce. Tím se otevírá možnost vytvářet unikátní vizualizace, které odpovídají specifickým požadavkům a designovým představám vývojářů.

Vega je velmi modulární a integruje se s dalšími populárními knihovnami a frameworky pro vizualizaci dat, jako je například D3. Spolu s těmito nástroji může Vega poskytnout ještě větší rozsah funkcí a možností pro pokročilé vizualizační efekty. Vegu je mimo jiné také možné využít pro vyobrazení vizualizace v rámci SVG, ale také za pomocí Canvas prvku. Díky podpoře interaktivity a animací je Vega také ideální pro vytváření dynamických a interaktivních vizualizací. Uživatelé mohou snadno přidávat interaktivní prvky, jako jsou filtry, výběry, přechody a další, které umožňují uživatelům prozkoumávat a manipulovat s daty.

Vega je tedy vysoce výkonná a škálovatelná knihovna, která podporuje tvorbu vizualizací pro různé platformy a aplikace. Je open-source projektem s aktivní komunitou vývojářů, kteří přispívají k jejímu rozvoji, opravám chyb a podpoře uživatelů. Pokud vám tedy nestačí jednodušší knihovny jako je například Metrics Graphics, tak je možné využít tuto knihovnu pro pokročilé vizualizace.

4.4.5 D3

D3 (Data-Driven Documents) je výkonná JavaScriptová knihovna určená pro manipulaci s dokumenty na základě dat. Je vyvinuta jako open-source projekt a její oficiální web nabízí rozsáhlou dokumentaci a příklady [14]. Tato knihovna umožňuje tvorbu dynamických a interaktivních vizualizací dat prostřednictvím použití standardních webových technologií, jako je HTML, CSS a SVG. Knihovna poskytuje silné nástroje pro selekci, manipulaci a transformaci document object model (DOM) na základě dat, což umožňuje vytvářet bohaté a poutavé vizualizace.

Jednou z klíčových vlastností D3 je jeho důraz na deklarativní přístup k tvorbě vizualizací. Uživatelé mohou popsat vzhled a chování vizualizace pomocí datově řízených transformací, které jsou aplikovány na objekty. Tímto způsobem lze snadno propojit data s vizuálními prvky a vytvářet reaktivní

vizualizace, které automaticky reagují na změny dat.

D3 je také známá pro svou flexibilitu a možnost vytvářet vlastní vizuální prvky a interakce. Knihovna poskytuje mnoho nástrojů a metod pro práci s grafikou, animacemi, změnou atributů, řízením událostí a mnoho dalšího. To umožňuje tvorbu unikátních a přizpůsobitelných vizualizací podle specifických požadavků.

D3 je široce využívána v oblasti datové vizualizace a je preferovanou volbou pro profesionální vývojáře a designéry. Její komunita je velmi aktivní a poskytuje podporu, rozšíření a další užitečné nástroje, které rozšiřují funkcionality knihovny. Jedná se o jednu z nejobsáhlejších JavaScriptových knihoven pro tvorbu vizualizací v tuto chvíli.

4.4.6 Chart.js

Chart.js je populární JavaScriptová knihovna pro tvorbu interaktivních a esteticky příjemných vizualizací na webových stránkách. Tato knihovna je vysoce konfigurovatelná a snadno použitelná, což umožňuje i začátečníkům rychle vytvářet poutavé vizualizace.

Dle oficiálního webu tato knihovna podporuje různé typy grafů, včetně sloupcových, spojnicových, koláčových, paprskových a mnoho dalších komunitně vytvořených grafů [15]. Uživatelé mohou snadno definovat data, popsat styly, nastavit osy a přidávat popisky, čímž vytvářejí vizualizaci přesně podle svých potřeb. Navíc pokud uživatel nijak neupraví data předaná do vizualizace, tak knihovna využije svého předem nastaveného vzhledu pro danou vizualizaci, který zahrnuje i defaultní animace, což může pomoct lépe předat požadovaný příběh.

Jednou z hlavních výhod Chart.js je i jeho responzivní design, který umožňuje, aby se vizualizace automaticky přizpůsobovaly velikosti a rozlišení obrazovky. Tímto způsobem se zajišťuje, že vizualizace zůstávají čitelné a esteticky příjemné na různých zařízeních, včetně mobilních telefonů a tabletů.

Chart.js poskytuje také možnosti interaktivity, které umožňují uživatelům prozkoumat data ve vizualizaci. To zahrnuje možnost najetím na jednotlivé body vizualizace získat podrobnosti a možnost přepínat mezi různými sadami dat. Navíc oproti jiným knihovnám Chart.js vykresluje své vizuali-

zace za pomocí Canvas, což sice znemožňuje využití CSS stylů, ale pomáhá při vizualizaci velkých datových sad a komplexních vizualizací.

Velká síla Chart.js spočívá také v jeho rozšířitelnosti. Knihovna nabízí mnoho pluginů a rozšíření, které umožňují přidávat další funkcionality a vlastnosti do grafů. Uživatelé tak mohou přizpůsobit grafy podle svých individuálních požadavků a vytvářet unikátní vizualizace.

Jako mnoho dalších projektů je Chart.js open-source projekt s aktivní komunitou vývojářů. To znamená, že komunita poskytuje jak dokumentaci s příklady kódu, tak i podporu pro uživatele. Jedná se v dnešní době o velmi populární knihovnu využívanou ve webovém vývoji, která nabízí rychlé a efektivní zobrazení dat ve formě vizualizací na webových stránkách.

5 Implementace

V této kapitole budou rozebrány jednotlivé knihovny v rámci jejich schopnosti vizualizovat data s množinou funkcionalit na typech grafů, které byly definovány v kapitole 4. Pokud knihovna neumožňuje funkcionalitu definovanou v kapitole 4, tak jsou tyto funkcionality vynechány. Pokud se lze aspoň nějak přiblížit těmto funkcionalitám, tak dané funkcionality jsou implementovány v rámci best effort přístupu. Všechny použité datové sady pro jednotlivé knihovny lze nalézt v příloze ve složce Data. Zhotovené programy k jednotlivým knihovnám lze nalézt v příloze ve složkách s jménem knihovny (například Vega).

5.1 Metrics Graphics

V rámci této knihovny nepracujeme s nejnovější verzí, ale s verzí 2.13.0. To je dáno nezprovozněním nejnovější verze knihovny a neexistence dokumentace k ní vytvořené. Tato knihovna neumožňuje vykreslit všechny typy grafů, které byly definovány v kapitole 4.2, proto byly některé funkcionality přesunuty k jiným typům grafů, kde mají také smysluplné využití.

5.1.1 Data

```
Year,Global_Sales
1983.0,10.96
1984.0,45.56
1985.0,49.95
1986.0,16.18
1987.0,11.95
1988.0,36.44
```

Listing 5.1: Ukázka zdrojových dat

V rámci načítání dat a následné vizualizaci je v knihovně Metrics Graphics nutno nejdřív využít nějakou externí knihovnu pro načítání dat z CSV souboru, jelikož takovou vlastnost knihovna nemá. Strukturu CSV souboru lze vidět na obrázku 5.1. Toho lze poměrně jednoduše dosáhnout za pomocí propojení knihovny z D3, která tuto funkcionalitu obsahuje. Jedná se o knihovnu D3 verze 4, jelikož novější verze není podporována.

```
d3.csv('../Data/nintendo2005+_with_ratings.csv', function(data
    ↪ ));
```

Listing 5.2: Načítání dat pomocí D3

Alternativně je možné využít i AJAX, kde za pomocí XML HTTP požadavku je možné CSV soubor načíst.

```
request.open('GET', 'data/total_Nintendo_copies_by_year.csv',
    ↪ true);
```

Listing 5.3: Načítání dat pomocí AJAX

Poté, co jsou uložená data přečtena ze souboru je nutné tato data v podobě stringu převést na pole objektů. K tomu je možné využít vlastní skript nebo další externí knihovnu. Pro tuto knihovnu byla využita externí knihovna Papaparse, jelikož umožňuje jednoduché parsování dat z CSV souborů za pomocí header parametru.

```
const parsedData = Papa.parse(csvData, { header: true }).data;
```

Listing 5.4: parsování dat za pomocí Papaparse

Takto získané pole dat bylo následně potřeba upravit, aby byla knihovna Metrics Graphics schopná ho zpracovat. Z tohoto důvodu se využilo JavaScriptové funkce map(), která umožnila namapovat jednotlivého hodnoty ve správném formátu k jednotlivým objektům uvnitř předaného pole.

Jelikož MetricsGraphics je knihovna převážně pro zpracování datových řad, tak bylo nutné při mapování dat upravit hodnoty proměnné roku v rámci každého objektu, jelikož knihovna očekává datum ve formátu YYYY-MM-DD, ale data jsou ve formátu stringu čísla s jedním desetinným místem viz 5.1. Bylo tedy nutné formát dat nejprve převést ze stringu na integer, čímž jsme se zbavili desetinného formátu uloženého roku. Následně k takto získanému roku jsme přidali defaultní měsíc a den (string "-01-01"), ve kterém budeme uvedená data vizualizovat. V rámci dalšího mapování stačilo pouze převést jednotlivé hodnoty pro uvedené časy na čísla, čímž jsme získali data ve formátu, který knihovna Metrics Graphics očekává.

```
let dateParsedData = parsedData.map(d => ({
    ...d,
    Year: parseInt(d.Year) + "-01-01",
    Sales: Number(d.Global_Sales)
}));
```

Listing 5.5: Mapování dat pro časové řady

Alternativně u scatter plotu knihovna očekává hodnoty ve formátu číslo/číslo místo datum/číslo, takže z načítané datové sady pro scatter plot bylo nutné pouze převést jednotlivé stringy hodnot na čísla.

```
[{Year:1983, Global_Sales:10.96},
{Year:1984, Global_Sales:45.56},
{Year:1985, Global_Sales:49.95},
{Year:1986, Global_Sales:16.18},
{Year:1987, Global_Sales:11.95},
{Year:1988, Global_Sales:36.44}]
```

Listing 5.6: Ukázka upravených dat

Na obrázku 5.6 lze vidět finální modifikovaná data, kde každý řádek je jeden objekt v poli objektů hodnot pro vizualizaci Nintendo tržeb. V rámci tohoto obrázku první řádek slouží pouze jako popisek dat.

Konfigurace V rámci vykreslení a konfigurace vizualizace používá Metrics Graphics jednotné volání metody knihovny data_graphic({konfigurace}). Konfigurace knihovny nám umožňuje přizpůsobení a vykreslení vizualizace podle našich představ. Nejpodstatnějšími atributy této konfigurace jsou položky:

1. data (určuje množinu dat, které chceme vykreslit)
2. x_accessor (jaká kategorie bude na ose X)
3. y_accessor (jaká kategorie bude na ose Y)
4. chart_type (určuje jaký typ grafu chceme zobrazit)

Bez těchto atributů totiž knihovna není schopna rozpoznat, co chceme vizualizovat, a tudíž nemůže nic vykreslit. Pokud tyto hlavní atributy nastavíme, knihovna využije své výchozí nastavení pro daný typ grafu a pomocí těchto atributů vykreslí celou vizualizaci. Při základním nastavení se

knihovna zobrazuje v relativně malém okně (220x350), což není ideální, pokud máme velké množství dat, která se rozprostírají na různých rozsazích převážně na ose X. Detailnější analýza takovýchto dat je poměrně problematická, jelikož knihovna vykresluje pouze omezené množství popisků na datových osách. Tento problém lze ale částečně omezit díky tooltipu, který knihovna zobrazuje v pravé části horního grafu. Za pomocí tohoto tooltipu je totiž možné zjistit hodnoty, kterých měřená sada dosahuje v daném roce i přes ne zcela jasné umístění na dané ose.

5.1.2 Spojnicový graf

Pro vyobrazení spojnicového grafu Metrics Graphics využívá hodnoty 'line' v rámci atributu chart_type. V rámci funkcionalit spojnicového grafu bylo vyzkoušeno i několik dodatečných funkcionalit z ostatních grafů, které knihovna Metrics Graphics nepodporuje.

Markery Jako první funkcionalitu se povedlo implementovat markery. Pro tuto funkcionalitu bylo nutné definovat nové pole objektů, do kterého byl požadovaný marker vložen. Markery musí být definovány ve tvaru 'jméno sloupce': hodnota, 'label': 'popisek markeru', kde jméno sloupce definuje, podle které osy se má marker vyobrazit a s jakou hodnotou. Hodnota musí odpovídat jednotkám, ve kterých se daná osa uvádí. Oproti tomu label určuje, jaký popisek se v rámci vizualizace zobrazí na grafu.

```
var markers = [{'Year': new Date('2000-01-01'), 'label': 'New
   ↪ management'},];
```

Listing 5.7: Ukázka pole markerů

Pro vložení markerů do vizualizace stačí pouze do konfigurace přidat atribut markers, ke kterému je přidělen objekt držící jednotlivé markery.

```
const chart = MG.data_graphic({
...
markers: markers,
...
}
```

Listing 5.8: Přidání markerů do konfigurace

Nastavení baseline V rámci dalších funkcionalit se podařilo naimplementovat baseline. Baseline se do grafu přidává stejným způsobem jako markery. Rozdíl spočívá v tom, že baseline je vždy pro osu Y, a tak se pouze zadává hodnota baseline a popisek, který se k ní má vyobrazit. V rámci přidávání do konfigurace je možné baseline přidat přímo bez předdefinování v nějaké proměnné držící pole baseline. Na obrázku 5.14 lze vidět formátování baseline a přímé přidání do konfigurace.

```
const chart = MG.data_graphic({
...
baselines: [{ value: 67.17, label: 'average of sold copies'
   ↪ },],
...
}
```

Listing 5.9: Přidání baseline do konfigurace

Popisky os Další funkcionalitou, která byla implementována, jsou popisky os. Popisky os se přidávají přímo v konfiguraci v atributu x_label a y_label, kde se jako hodnota přidá jméno pro danou osu ve tvaru 'jméno'. Dodatečné popisky os implementovat nelze.

```
const chart = MG.data_graphic({
...
x_label: 'Roky',
y_label: 'Prodeje (mil.)',
...
}
```

Listing 5.10: Přidání popisků do konfigurace

Výplň datového setu Výplně datové sady lze dosáhnout jednoduchých přidáním atributu area s hodnotou true. Pokud uživatel zapomene tento atribut definovat, tak se stejně vykreslí výplň, jelikož se jedná o základní nastavení pro spojnicový graf.

```
const chart = MG.data_graphic({
...
area: true,
...
}
```

Listing 5.11: Přidání popisků do konfigurace

Definování vlastní barvy Jako další implementovaná funkcionalita je definování vlastní barvy pro jednotlivé sady dat. Knihovna Metrics Graphics umožňuje definovat barvy za pomocí hex, rgba a jména dané barvy. Ve využívané verzi knihovny se vyskytla chyba, která zapříčinila nefunkčnost barevnost v rámci bar chartu.

```
var colors: ['tomato', '#CCFFFF']
```

Listing 5.12: Tvorba množiny barev

V rámci konfigurace je nutné pouze vložit pole hodnot pro jednotlivé datové sady do atributu color.

```
const chart = MG.data_graphic({
...
data: [[{y:...}...],[{y:...}]],
color: colors,
...
}
```

Listing 5.13: Přidání barev do konfigurace

5.1.3 Sloupcový graf

Pro vyobrazení sloupcového grafu Metrics Graphics využívá hodnoty 'bar' v rámci atributu chart_type.

Interaktivita grafů První implementovanou funkcionalitou v rámci sloupcového grafu je interaktivita grafu. V Metrics Graphics lze dosáhnout interaktivity za pomocí atributu mouseover, mouseout nebo mousemove. Do hodnoty těchto atributů lze definovat JavaScriptová funkce, která se provede po definované interakci s grafem. Do těchto funkcí lze vložit libovolný kód.

```
const chart = MG.data_graphic({
...
mousemove: mousemove_function(this.value),
mouseout: function(){
...
},
mouseover: mouseover_function(something),
...
}
```

Listing 5.14: Konfigurace interakcí

Škálovatelnost os Škálovatelnosti os lze v Metrics Graphics dosáhnout za pomocí atributu y_scale_type a x_scale_type. K těmto atributům stačí přiřadit hodnota jako 'linear' a daná osa se upraví na požadovaný tvar.

```
const chart = MG.data_graphic({
...
y_scale_type: 'linear',
x_scale_type: 'linear',
...
}
```

Listing 5.15: Úprava škálovatelnosti os v konfiguraci

5.1.4 Scatter plot

Pro vyobrazení scatter plot grafu Metrics Graphics využívá hodnoty 'scatter' v rámci atributu chart_type. Scatter plot používá oproti ostatním grafům v rámci Metrics Graphics data ve tvaru číslo/číslo, jak bylo již zmíněno dříve. Z tohoto důvodu se využila nová datová sada viz obrázek 5.16.

```
Name,User_Score,Critic_Score
Mario Kart DS,8.6,91.0
Brain Age: Train Your Brain in Minutes a Day,7.9,77.0
Brain Age 2: More Training in Minutes a Day,7.1,77.0
Animal Crossing: Wild World,8.7,86.0
Big Brain Academy,7.4,74.0
Pokemon Mystery Dungeon: Blue Rescue Team,8.1,62.0
Super Mario Strikers,8.7,76.0
```

Listing 5.16: Příklad dat pro scatter plot

Posunutí báze Posunutí báze lze v rámci konfigurace knihovny dosáhnout poměrně jednoduše. Pro posunutí báze se využívá atributu min_y, ke kterému se přiřadí hodnota, která odpovídá hodnotám na ose Y.

```
const chart = MG.data_graphic({
...
min_y:53,
...
}
```

Listing 5.17: Konfigurace posunutí báze grafu

Modifikace dat V rámci modifikace dat Metrics Graphics lze implementovat velikost bodů. Toho se docílí přidání atributu point_size přímo do konfigurace grafu, ke kterému se přiřadí hodnota, která definuje celkovou velikost bodu na grafu

```
const chart = MG.data_graphic({
...
point_size: 4,
...
}
```

Listing 5.18: Konfigurace posunutí báze grafu

5.2 Google Charts

V rámci implementace Google Charts se využívá nejaktuálnější verze (52). Tato knihovna podobně jako knihovna Metrics Graphics neumožňuje zobrazit všechny definované typy grafů. Z toho důvodu jsou některé funkcionality přesunuty k jiným typům grafů, kde dává smysl jejich použití.

5.2.1 Data

```
Year,Nintendo,Sony,Microsoft,EA,Activison,Prediction
1983,10.96,0.0,0.0,0.0,1.94,10.0
1984,45.56,0.0,0.0,0.0,0.27,38.5
1985,49.95,0.0,0.0,0.0,0.48,50.0
1986,16.18,0.0,0.0,0.0,0.0,35.0
1987,11.95,0.0,0.0,0.0,1.12,14.0
1988,36.44,0.0,0.0,0.0,0.75,21.0
```

Listing 5.19: Ukázka multikategorických zdrojových dat

V rámci načítání dat a jejich následné vizualizaci v knihovně Google Chart je nutné využít externí knihovnu pro načítání dat z CSV souboru. Stejně jako tedy v rámci kapitoly 5.1.1 jsme využili knihovnu D3. Jediným rozdílem je, že se jedná o aktuální verzi knihovny (7.1.1) a nepoužili jsme Papaparser.

Oproti většině knihoven Google Charts je velmi specifická, co se týče formátu dat. Google Charts využívá tzv. datatables, což jsou speciální objekty, které drží data pro vizualizaci.

```
var googleChartsData = new google.visualization.DataTable();
```

Listing 5.20: Příklad deklarace DataTable

Specifické na těchto objektech je potřeba nejprve definovat jednotlivé sloupce i s typem dat, která daná datová sada bude obsahovat a teprve následně načítat data bez jejich headerů. Mezi typy sloupců, které se dají takto definovat, patří i několik speciálních typů, které určují, že se jedná o tzv. "vlastnosti" daných dat. Přesněji se jedná o možnosti, jak přidat k jednotlivým datům anotace, barvy a další viz obrázek 5.21.

```
googleChartsData.addColumn('string', 'Year');
googleChartsData.addColumn('number', 'Nintendo');
googleChartsData.addColumn({type: 'string', role: 'annotation
    ↪ '})
googleChartsData.addColumn({type: 'string', role: '
    ↪ annotationText'})
```

Listing 5.21: Příklad deklarace sloupců v rámci DataTable

Poté, co jsou takto definovány sloupce pro jednotlivé hodnoty, které se budou ukládat v DataTable, je na řadě samotné formátování dat. Pro formátování dat je následně nutné využít knihovní funkce addRow() v rámci vlastního skriptu, který přečtená data jednotlivě projde a přidá je po řádcích do DataTable.

```
data.forEach(function(row) {
    googleChartsData.addRow([
    row.Year,
    parseFloat(row.Nintendo),
    null,
    null
    ]);
});
```

Listing 5.22: Příklad vkládání dat

Jednotlivé buňky v rámci DataTable jsou na sobě částečně nezávislé, a proto je možné jejich hodnoty aktualizovat v rámci definovaných datových typů (string, integer, ...) viz obrázek 5.23.

```
googleChartsData.setValue(23, 2, 'New Super Mario Bros.');
googleChartsData.setValue(23, 3, 'Most selling title of the
    ↪ period');
googleChartsData.setValue(33, 2, 'Kirby');
googleChartsData.setValue(33, 3, 'Worst selling title of the
    ↪ period');
```

Listing 5.23: Příklad aktualizace buněk DataTable

Konfigurace K vykreslení požadované vizualizace je nutné nejprve vytvořit instanci daného grafu s předaným parametrem určující, do čeho se má tato vizualizace vyobrazit. Následně je nutné zavolat knihovní funkci draw (data, konfigurace).

```
var optionsLine = {
...
}

var chart = new google.visualization.LineChart(document.
    ↪ getElementById('myChart'));
chart.draw(googleChartsData,optionsLine);
```

Listing 5.24: Vykreslení objektu s konfigurací

K předání konfigurace vizualizace se využívá objekt, který obsahuje všechna požadovaná nastavení. Díky tomu, že v rámci DataTable jsou přesně definované sloupce, které se vykreslí na ose X a Y, tak tato knihovna po deklarování chtěného typu grafu ho vykreslí i bez definování jakékoliv konfigurace. Knihovna Google Charts se v tomto případě snaží vyobrazit všechny sloupce v DataTable, což zahrnuje i speciální atributy, jako je anotace, barva atd. K tomuto zobrazení využije své defaultní nastavení, které je poměrně dostatečné i pro větší datové sady. Defaultní nastavení obsahuje i automatické přidání hover efektu pro tooltip, který vypíše hodnoty daných bodů.

5.2.2 Spojnicový graf

Pro vykreslení spojnicového grafu v Google Charts se využívá metody LineChart(element), kde element je oblast, do které se daný graf má vykreslit.

Úprava typů čar v grafu Jako první funkcionalitu se povedlo implementovat úpravu typů čar. Pro přidání konfigurace pro úpravu čar je potřeba pouze vložit atribut jako curveType, lineDashStyle nebo lineWidth přímo do konfigurace daného grafu.

```
var optionsLine = {
...
curveType: 'function',
lineDashStyle: [10,1],
lineWidth:3,
...
}
```

Listing 5.25: Přidání úpravy typů čar do konfigurace

Nastavení baseline Nastavení base line v Google Charts není zcela přímočaré. Nejdřív je totiž nutné uvnitř konfigurace přidat tzv. series, což jsou vlastně série jednotlivých dat. Následně je potřeba vytvořit další sérii k té původní, která bude obsahovat pouze baseline, a její nastavení společně s místem, kde se má vykreslit.

```
var optionsLine = {
...
vAxes: {
    0: {title:'Global sales',},
    1: {
        textStyle: {
            color: 'red'
        },
        baseline:67,
        baselineColor: 'red',
        ticks: [67]
        }
}
...
}
```

Listing 5.26: Příklad přidání baseline k ose Y

Slicing Slicing lze implementovat v knihovně Google Charts poměrně jednoduše. Stačí pouze definovat viewWindow k požadované ose, což nám zobrazí data pouze v definovaném rozmezí a ostatní data nevykreslí.

```
var optionsLine = {
...
vAxis: {
    viewWindow: {
      min: 0,
      max: 250
    }
},
...
}
```

Listing 5.27: Příklad konfigurace slicingu

Markery a dodatečné popisky Markery a dodatečné popisky lze přidat do grafů pouze při načítání dat, kde je sloupec definován jako annotation viz obrázek 5.21 a 5.23. To znamená, že jednotlivé popisky lze přidat pouze k jednotlivým bodům a nikoliv kdekoli na grafu. V konfiguraci grafu je následně možnost upřesnění stylu těchto bodů za pomocí style atributu v rámci annotations.

```
var optionsLine = {
...
annotations: {
    style: 'point',
},
...
}
```

Listing 5.28: Příklad konfigurace anotace

Popisky os a stylizování os Popisky a stylizaci os lze upravovat přímo v konfiguraci za pomocí atributu hAxis nebo vAxis. K těmto atributům lze následně definovat další konfiguraci.

```
var optionsLine = {
hAxis: {
    title: 'Years',
    ... },
}
```

Listing 5.29: Příklad konfigurace popisků osy X

5.2.3 Sloupcový graf

Pro vykreslení sloupcového grafu v Google Charts se využívá metody BarChart(element), kde element je oblast, do které se daný graf má vykreslit.

Interaktivita grafů Google Charts v rámci interaktivity grafu nabízí možnosti eventů. Eventy umožňují zachytit jednotlivě akce prováděné na grafu (kliknutí na sloupec, označení textu, ...) a následně při nich spustit definovanou funkci.

Nejdříve je potřeba přidat odposlech jednotlivých akcí (např. select), ke kterému je definována i funkce, která se má následně provést. Co je možné

v této funkci napsat, je pouze v představivosti uživatele, ale především se jedná o vypisování hlášek webu nebo podobných příkazů.

```
function selectHandler() {
    var selectedItem = chart.getSelection()[0];
    if (selectedItem) {
        var year = data.getValue(selectedItem.row, 0);
        alert('The user selected year: ' + year);
    }
}
google.visualization.events.addListener(chart, 'select',
    ↪ selectHandler);
}
```

Listing 5.30: Příklad vypsaní roku na obrazovku po kliknutí

Definování vlastní barvy Definovat vlastní barvu lze v této knihovně pro jednotlivé osy a nebo data. Barvy se dají definovat za pomoci obecně známých formátů pro webově prostředí jako je hex, rgb, názvem barvy nebo cmyk.

```
['red','#32a852','rgb(50,168,82), ...]
```

Listing 5.31: Příklad definování barvy

Předdefinované sady barev Google Charts nabízí pouze jedu předdefinovanou sadu barev pro svoji vizualizaci. Pokud tuto sadu vyčerpá, tak je následně znovu použita první barva této sady.

```
['#3366cc','#dc3912','#ff9900','#109618',
'#990099','#0099c6','#dd4477','#66aa00',
'#b82e2e','#316395','#994499','#22aa99',
'#aaaa11','#6633cc','#e67300','#8b0707',
'#651067','#329262','#5574a6','#3b3eac',
'#b77322','#16d620','#b91383','#f4359e',
'#9c5935','#a9c413','#2a778d','#668d1c',
'#bea413','#0c5922','#743411']
```

Listing 5.32: Celá sada barev pro Google Charts

Škálovatelnost os Pro implementaci škálovatelnosti os v rámci knihovny Google Chart je potřeba pouze upravit konfiguraci v rámci vybrané osy (např. hAxis). V této úpravě je nutné definovat atribut scaleType a přidat k němu požadovanou hodnotu.

```
var optionsLine = {
...
hAxis:: {
    scaleType: 'log',
},
...
}
```

Listing 5.33: Příklad logaritmické škály

Stacked a stacked 100% Pro získání stackable a 100% sloupcového grafu je potřeba v konfiguraci přidat atribut isStacked. Pro definování stackable grafu je potřeba přidat hodnota true a pro 100% stackable hodnota 'percent'.

```
var optionsLine = {
...
isStacked: 'percent',
...
}
```

Listing 5.34: Příklad konfigurace 100% stackable grafu

5.2.4 Scatter plot

Pro vykreslení spojnicového grafu v Google Charts se využívá metody BarChart(element), kde element je oblast, do které se daný graf má vykreslit.

Modifikace dat Modifikace dat z pohledu jejich vzhledu jde v knihovně implementovat poměrně jednoduše. Do již vytvořené konfigurace se přidají další elementy jako pointSize nebo pointShape a je k nim přiřazena požadovaná hodnota.

```
var optionsLine = {
...
pointShape: 'diamond',
...
}
```

Listing 5.35: Příklad změny vizuálu bodů grafu

Posunutí báze K implementaci posunutí báze se využívá atribut viewWindow v rámci konfigurace, který osu posune na požadovanou hodnotu. Příklad konfigurace je na obrázku 5.27.

Filtrace dat Přímá filtrace dat v knihovně není implementována, ale její "light verze" lze docílit za pomocí slicingu. Slicing totiž ostatní data mimo okno ořeže a docílí jakéhosi vyobrazení "filtrace". Konfigurace je stejná jako na obrázku 5.27.

5.3 D3

V rámci implementace D3 se využívá verze d3v6. Tato knihovna umožňuje zobrazit všechny definované typy grafů, jelikož se jedná v aktuální chvíli o nejobsáhlejší knihovnu.

5.3.1 Data

V rámci části zpracování dat se využívá stejných funkcí jako u dat Metrics Graphics, jelikož Metrics Graphics využívá právě tuto knihovnu viz obrázek 5.2. Data mají také stejný charakter jako data uvedená v kapitole 5.2.1 o Google Charts. Mapování načtených hodnot za pomocí metody csv() probíhá také zcela totožně s knihovnou Metrics Graphics.

Konfigurace Zásadní rozdíl mezi ostatními knihovnami spočívá ve způsobu, jak jsou do grafu přidávány jednotlivé komponenty. V D3 totiž neexistuje nic ve stylu chart_type: 'typ grafu', jelikož samotný graf se vytváří ze skládání různých prvků knihovny do jednoho vizualizačního okna. Není tedy možné definovat defaultní konfiguraci, jelikož knihovna sama neví, jaký typ grafu chce uživatel vykreslit.

Jediné, co knihovna opravdu dobře zná, jsou načtená data a objekty, které je možné různě skládat do zobrazovaného okna. Tato vlastnost přináší obrovské množství konfiguračních možností pro graf, ale zároveň vytváří problémy s vysokou složitostí používání této knihovny.

Mezi nejdůležitější objekty knihovny v rámci vykresování dat patří:

1. Text - objekt pro text
2. Line - objekt linie
3. Rect - objekt pro sloupce
4. Path - objekt cesty

5.3.2 Spojnicový graf

Spojnicový graf se převážně implementuje za pomocí line a path objektu.

Úprava typů čar v grafu Úprava typů čar probíhá při přidání cesty pro vygenerovanou čáru. Modifikování čar se provádí v rámci atributů dané čáry ve tvaru ("co", "jak"). "Co" znamená, jaký atribut dané čáry chceme upravit, a "jak" určuje, co se do daného atributu má nastavit za hodnotu.

```
...
const path = svg.append("path")
    .datum(data)
    .attr("fill", "none")
    .attr("stroke", "steelblue")
    .attr("stroke-width", 1.5)
    .attr("stroke-dasharray", "[5,5]")
    .attr("d", line);
...
}
```

Listing 5.36: Příklad úpravy stylu linií

Nastavení baseline Baseline se implementuje za pomocí přidání objektu typu "line", kterému se v atributech přiřadí, kde se má na osách vykreslit, s jakou tloušťkou, barvou, atd.

```
...
svg.append("line")
    .attr("class", "baseline")
    .attr("x1", marginLeft)
    .attr("y1", y(67))
    .attr("x2", width - marginRight)
    .attr("y2", y(67))
    .attr("stroke", "red")
    .attr("stroke-width", 1);
...
}
```

Listing 5.37: Příklad přidání baseline

Slicing Slicing lze v knihovně D3 implementovat za pomocí nastavení domény vykreslovaných dat. Na grafu se totiž zobrazují pouze hodnoty, které jsou v doméně definovány, což umožňuje poměrně jednoduché zobrazení pouze požadovaných data.

```
x.domain(d3.extent(filteredData, d => d.Year));
```

Listing 5.38: Příklad slicingu na ose X

Dodatečné popisky Dodatečné popisky lze implementovat k jakémukoliv objektu a na jakékoliv místo v grafu. Problém se staticky definovaným místem je ale v tom, že se poměrně těžko vypočítává požadovaná poloha.

```
svg.append("text")
   .attr("class", "axis-label")
   .attr("x", width / 2 + 45)
   .attr("y", height - 10)
   .text("Peak")
   .attr("text-anchor", "middle");
```

Listing 5.39: Příklad přidání dodatečných popisků

5.3.3 Sloupcový graf

Sloupcový graf se převážně implementuje za pomocí rect objektu.

Interaktivita grafů Interaktivitu grafu lze implementovat za pomocí přidání události na každý sloupec v grafu. K tomuto slouží metoda on ("událost", akce) při tvorbě sloupců, která umožňuje definovat události v rámci grafu a přidělit k nim funkci, která se má vykonat.

```
.on("click", function(d) {
  var year = d3.select(this).attr("data-year");
  alert("Year:" + year);
})
```

Listing 5.40: Příklad přidání interakce po kliknutí

Definování vlastní barvy Definovat vlastní barvu v D3 lze ke každému objektu samostatně a barva lze definovat pomocí hex, rgb, rgba nebo jména barvy (string). Barva každého objektu se definuje za pomocí úpravy hodnot jeho atributu (fill nebo stroke) viz obrázek 5.37.

Škálovatelnost os Pro škálovatelnost os v knihovně D3 je nutné vytvořit proměnnou dané škály a na ní následně zavolat metodu scaleLog (nebo jiný typ škálování) společně s namapováním domény a rozsahu. Takto vytvořené škálováni je následně potřeba pouze vložit do volání pro tvorbu dané osy.

```
var xScale = d3
    .scaleBand()
    .domain(data.map(function(d) { return d.Year; }))
    .range([0, innerWidth])
    .padding(0.1);
```

Listing 5.41: Příklad definice škálovaní osy

Stacked a stacked 100% Pro implementaci Stacked a 100% stacked je nutné provést ještě transformaci načítaných dat podle klíče, jelikož v rámci jednoho sloupce chceme zobrazit několik kategorií zároveň.

```
var stackedData = d3.stack().keys(categories).offset(d3.
    ↪ stackOffsetExpand)(data);
```

Listing 5.42: Transformace dat pro stack formát

Je také nutné ke každé skupině přiřadit barvu výplně podle klíče, aby bylo možné následně jednotlivé sloupce vykreslit.

```
var groups = svg.selectAll("g.group")
    .data(stackedData)
    .enter()
    .append("g")
    .attr("class", "group")
    .style("fill", function(d) { return colors(d.key); });

var rects = groups.selectAll("rect")
    .data(function(d) { return d; })
    .enter()
    .append("rect")
    .attr("x", function(d) { return xScale(d.data.Year); })
    .attr("y", function(d) { return yScale(d[1]); })
    .attr("height", function(d) { return yScale(d[0]) - yScale(
        ↪ d[1]); })
    .attr("width", xScale.bandwidth());
```

Listing 5.43: Příklad vykreslení 100% stackable sloupců

5.3.4 Scatter plot

Scatter plot se převážně implementuje za pomocí path objektu.

Modifikace dat Modifikaci dat možné uskutečni v rámci přidávání bodů na graf. Jedná se především o modifikaci ohledně tvaru a výplně, kterou budou jednotlivé datové body obsahovat. Všechny vlastnosti jednotlivých bodů se definují pomocí volání metody attr ("co", "jak"), kde se určí co a jak se má přizpůsobit viz kapitola 5.3.2.

```
svg.selectAll("path")
        .data(data)
        .enter()
        .append("path")
        .attr("d", diamond)
        .attr("fill", "steelblue");
```

Listing 5.44: Příklad změny tvaru dat

Posunutí báze Posunutí báze lze dosáhnout konfigurací objektu pro škálování dané osy, kde v doméně definujete vaší požadovanou hodnotu ve tvaru [min, max].

```
var yScale = d3.scaleLinear()
    .domain([50, d3.max(data, function(d) { return d.
       ↪ Critic_Score; })])
    .range([height - margin, margin]);
```

Listing 5.45: Příklad posunutí báze Y

Filtrace dat Pro filtraci dat lze použít metodu filter ("podmínka"), která prohledá data a do množiny uvede jen ta, která splňují předanou podmínku.

```
const filteredData = data.filter(d => d.Year >= new Date(
   ↪ startYear, 0, 1) && d.Year <= new Date(endYear, 0, 1));
```

Listing 5.46: Příklad filtrace dat na základě roku

5.3.5 Paprskový graf

Pro paprskový graf se využívá kombinace všech hlavních objektů knihovny.

Stylizování os grafu Knihovna D3 umožňuje poměrně bohatou stylizaci pro jednotlivé osy grafu. Tato stylizace se definuje v rámci vykreslení os, kde jako do atributů jednotlivých os jsou vkládány požadavky na osy grafu. Z pohledu stylizace popisků jednotlivých os je potřeba tuto stylizaci provést při jejich tvorbě stejným formátem atributů jako v případě os.

```
.attr("stroke", "gray")
.attr("stroke-width", 1)
.attr("opacity", 0.3)
```

Listing 5.47: Příklad modifikace os

Popisky os Popisky os se vkládají jako samostatné objekty typu text. Těmto objektům je nutné definovat místo, na které se daný objekt má vykreslit, a je zde možné definovat styl a atributy daného textu, jak bylo zmíněno výše.

```
.attr("text-anchor", "middle")
.style("font-weight", 600)
```

Listing 5.48: Příklad modifikace popisků

Vyplň datové sady Výplň datového sady lze implementovat za pomocí atributů v rámci vlastnosti jednotlivých objektů viz kapitola 5.36.

5.3.6 Bullet chart

Bullet chart se implementuje převážně pomocí rect a path objektů.

Markery Markery v rámci knihovny D3 jde implementovat za pomocí objektů linie a textu. Způsob vykreslení daného markeru zcela záleží na typu vizualizace, kterou uživatel chce vyobrazit. V případě markerů pro bullet chart se jedná o implementaci linie v rámci jednotlivých sloupců. V rámci implementace byly markery vytvořeny bez popisků, jelikož markery jsou zahrnuty v legendě.

```
groups.append("line")
    .attr("x1", function(d, i) { return xScales[i](d.markers
       ↪ [0]); })
    .attr("y1", 0)
    .attr("x2", function(d, i) { return xScales[i](d.markers
       ↪ [0]); })
    .attr("y2", 20)
    .style("stroke", "#000000")
    .style("stroke-width", 2);
```

Listing 5.49: Příklad deklarace markerů na ose

Modifikace legendy V rámci modifikace a tvorby je nejprve potřeba definovat objekt, do kterého se legenda bude vykreslovat. Následně je potřeba přidat objekty pro dané věci, které chceme v legendě zobrazit. V rámci implementace jsme využili objektu rect pro předání informace o barvě položky a objekt text pro přidání jména položky.

```
legendItems.append("rect")
     .attr("x", 0)
     .attr("y", 5)
     .attr("width", 10)
     .attr("height", 10)
     .style("fill", function(d) { return d.color; });
```

Listing 5.50: Příklad přidání barev legendy

```
legendItems.append("text")
     .attr("x", 20)
     .attr("y", 10)
     .attr("dy", "0.35em")
     .style("font-size", "12px")
     .text(function(d) { return d.label; });
```

Listing 5.51: Příklad přidání popisků legendy

Předdefinované sady barev Knihovna D3 má mnoho předdefinovaných sad barev, které je možné využít za pomocí volání d3.schemeCategory + číslo, kde číslo udává definovanou sadu barev.

```
var colorScale = d3.scaleOrdinal(d3.schemeCategory10);
```

Listing 5.52: Příklad využití barevných sad

5.4 Vega

V rámci implementace využíváme knihovny Vega verze 5.25.0, Vega-lite verze 5.9.1 a Vega-embed verze 6.22.1. Tato knihovna umožňuje zobrazit všechny definované typy grafů, jelikož se jedná o velmi pokročilou knihovnu, která je schopna využívat i funkce D3.

5.4.1 Data

Tato knihovna obsahuje speciální způsob pro čtení dat z CSV souborů. Místo běžného načítání za pomocí D3 nebo externí knihovny je zde souborová cesta vložena do atributu konfigurace url. Knihovna následně sama přečte daný soubor a načte si data do požadované struktury. Poté, co knihovna takto přečtená data uloží, je možné data transformovat do chtěné podoby za pomoci knihovních funkcí. Vstupní data mají stejný charakter jako data uvedená v kapitole 5.2.1 o Google Charts.

```
var config = {
    data: {
        url: "../Data/Rok/nintendo2005+_with_ratings.csv",
    },
}
```

Listing 5.53: Příklad načítání dat

Konfigurace K vykreslení dané vizualizace je nutné vytvořit instanci grafu (v rámci něhož je definovaná i jeho konfigurace) a zavolat knihovní funkci vegaEmbed('#vis', graf), kde #vis je DOM objekt, do kterého se daná vizualizace bude vyobrazovat, a graf je instance daného grafu (jeho config).

Konfigurace je podobná knihovně Google Charts jen s tím rozdílem, že je složitější. To je především dáno větším množstvím možných úprav, než bylo možné v knihovně Google Charts. Knihovna pro určení typu grafu využívá atribut mark. Tento atribut nabývá různých hodnot, ale nejzákladnějšími hodnotami jsou 'line', 'category', 'point' a 'bar'.

5.4.2 Spojnicový graf

Pro vyobrazení spojnicového grafu Vega využívá hodnoty 'line' v rámci atributu mark.

Úprava typů čar v grafu Jako první funkcionalitu se povedlo implementovat změnu typu linie v rámci grafu. Pro tuto funkcionalitu je nutné v konfiguraci vložit do atributu mark objekt, který sice také je typu line, ale k němu je nutné ještě definovat atribut strokeDash, který nám určuje, jak má vypadat přerušovaná linie.

```
mark: {
  type: "line",
  interpolate: "natural",
  strokeDash: [4, 2]
},
```

Listing 5.54: Příklad rozšíření atributu mark

Nastavení baseline Pro nastavení baseline bylo potřeba nejdříve pomocí transformace vytvořit novou proměnnou s hodnotou baseline a uložit ji.

```
transform: [{ calculate: "67", as: "baseline" },]
```

Listing 5.55: Vytvoření baseline hodnoty

Poté, co byla hodnota uložena, bylo následně potřeba využít atributu layer. Tento atribut se dá přirovnat atributu series v knihovně Google Charts. V rámci tohoto atributu bylo potřeba definovat další marker, který musel být nastaven na typ 'rule'. Tímto bylo možné vytvořit baseline, kterou za pomocí další specifikace bylo možné upravovat.

```
 mark: 'rule',
encoding: {
    y: {field: "baseline", type: "quantitative"},
    color: {value: "red"},
    size: {value: 1}
}
```

Listing 5.56: Vyobrazení baseline hodnoty

Slicing a filtrace dat Knihovna nemá přímo metodu, jak dosáhnout slicingu, ale lze využít filtraci dat pro dosažení podobného výsledku. Jedná se tedy o stejnou funkci jako filtrace dat. To znamená, že pro dosažení efektu slicingu je nutné využít transformace dat, kde jednotlivá data jsou filtrována podle definované podmínky. Takto zpracovaná data jsou následně zobrazena a vytvářejí efektu slicingu.

```
transform: [
    ...
    { filter: "(datum.Year>=2000)&&(datum.Year<=2010)" },
    ...
],
```

Listing 5.57: Příklad filtrace/slicingu dat

Dodatečné popisky a markery Dodatečné popisky je možné vkládat do grafu v rámci atributu layer s textovým typem atributu mark. Poté, co je tento mark vytvořen a stylizován, je nutné za pomocí atributu encoding určit, kde se má popisek zobrazit. V rámci tohoto atributu je nutno definovat pozici na ose x, y a text, který se má vykreslit. Markery se vykreslují obdobným způsobem jako dodatečné popisky. Jediným rozdílem je nutnost vykreslit dodatečný bod/linii, ke které se popisek vykreslí.

```
mark: {
  type: "text",
  fontSize: 12
},
encoding: {
  x: {value: 1},
  y: {value: 56},
  text: "dodatecny popisek"
}
```

Listing 5.58: Příklad přidání dodatečného popisku

5.4.3 Sloupcový graf

Pro vyobrazení sloupců Vega využívá hodnoty 'bar' v rámci atributu mark.

Interaktivita grafů Další implementovanou funkcionalitou byla interaktivita grafů. Zde se povedlo za pomocí tooltipu definovat vlastní zobrazení při označení sloupce. K tomu bylo nutné v rámci atributu encoding definovat atribut tooltip, ke kterému následně byly přidány objekty obsahující informace, které měly být zobrazeny.

```
encoding: {
    ...
    tooltip: [
        { field: 'Category', title: 'Company' },
        { field: 'value', title: 'Value' },
        {field: 'rounded_percent', title:'%'}
    ],
    ...
}
```

Listing 5.59: Příklad přidání tooltipu

Definování vlastní barvy Definovat vlastní barvu lze v knihovně Vega za pomocí hex, rgb, rgba nebo jména barvy. Barva každé části vizualizace se upravuje za pomocí dodatečného atributu color v atributu encoding, kde se pro každý objekt přiřadí definovaná barva.

```
encoding: {
    ...
    color: ["yellow", "#32a852", "rgb(50,168,82)", ...]
    ...
}
```

Listing 5.60: Příklad definování barvy

Škálovatelnost os Změny škálovatelnosti os lze dosáhnout pomocí atributu přidaného atributu scale do atributu pro danou osu, čímž se jednoduše změní škálovatelnost pro požadovanou osu.

```
y: {
    ...
    scale: {
        type: 'log'
    },
    ...
}
```

Listing 5.61: Příklad definování logaritmického škálování

Stacked a stacked 100% Pro implementaci Stacked a stacked 100% grafu je potřeba využít transformace podobně jako v knihovně D3. Nejdříve je nutno složit jednotlivé kategorie do jednoho objektu ve tvaru ["Category", "Value"]. Následně je potřeba využít klíčování podle roku, kde se jednotlivé hodnoty spojí a podle těchto spojených hodnot se vypočítá, jakou část sloupce by měla zahrnovat hodnota dané kategorie. Příklad této modifikace lze nalézt v souboru 100%StackableBarChart.html.

5.4.4 Scatter plot

Pro vyobrazení bodů Vega využívá hodnoty 'point' v rámci atributu mark.

Modifikace dat Modifikace dat lze docílit úpravou konfigurace pro atribut mark, kde se tento atribut rozšíří o další atributy jako je filled, shape, size a další.

```
mark: {
    type: "point",
    filled: true,
    shape: 'diamond',
    size: 100
},
```

Listing 5.62: Příklad modifikace dat

Posunutí báze Pro posunutí báze lze docílit konfigurací atributu scale pro jednotlivé osy. V tomto atributu je nutné definovat další atribut domain, která stejně jako v D3 definuje rozmezí, ve kterém se má daná osa vykreslit.

```
encoding: {
    x: {
        field: "User_Score",
        type: "quantitative",
        scale:{
            domain: [5,9.25],
        },
    },
...
}
```

Listing 5.63: Posunutí báze X

5.4.5 Paprskový graf

Pro tvorbu radar chartu se využívají kombinace typů hodnot mark.

Stylizování os grafu Stylizování os grafu se implementuje do vizualizace za použití rozšíření encode, kde se pro atribut enter nastaví všechny potřebné styly jako stroke, interpolate, x, y, strokeWidth a další.

```
encode: {
    enter: {
      interpolate: { value: "linear-closed" },
      x: { field: "x2" },
      y: { field: "y2" },
      stroke: { value: "lightgray" },
      strokeWidth: { value: 3 }
    }
}
```

Listing 5.64: Příklad stylizování os grafu

Popisky os Popisky os lze implementovat a modifikovat stejně, jako bylo uvedeno u paragrafu dodatečných popisků.

Výplň datové sady Výplň datové sady lze získat za pomocí modifikace konfiguračního souboru, kde pro vybraný object je potřeba nastavit atributy v enter, což jsou atributy fill a fillOpacity.

```
encode: {
    enter: {
        ...
        fill: {
            scale: "color",
            field: "category"
        },
      fillOpacity: { value: 0.2 }
    }
  }
```

Listing 5.65: Příklad přidání výplně

5.4.6 Bullet chart

Markery Implementace markerů je ve Vega poměrně jednoduchá a podobná jako v D3. Je totiž nutné využití atributu layer v konfiguraci, kde se přidá nová vrstva mark, která bude typu "tick", což je zmenšenina standardní line.

```
layer: [
{
    mark: {type: "tick"},
    encoding: {x: {field: "markers[0]"}, color: {value: "black"
        ↪ }}
}]
```

Listing 5.66: Příklad přidání markeru

Modifikace legendy Modifikovat legendu jde v konfiguraci za pomocí atributu color uvnitř atributu encoding. Tento atribut nám umožňuje přidat jednotlivé barvy k popiskům legendy, jde zde nastavit velikost fontu, jméno legendy a další vlastnosti.

```
encoding: {
    color: {
        field: "color",
        type: "nominal",
        scale: {
            domain: ["2007", "2006", "Expectations"],
            range: ["steelblue", "lightsteelblue", "black"]
        },
        legend: {
            title: "Measures",
            labelFontSize: 12
        }
    }
}
```

Listing 5.67: Příklad modifikace legendy

5.5 ChartJS

V rámci implementace ChartJS se využívá nejaktuálnější verze 4.3.0 a rozšíření na anotaci pro tuto verzi knihovny.

5.5.1 Data

Knihovna načítá data ve stejném formátu jako knihovna Google Charts viz obrázek 5.19. Pro načítání těchto dat byla zvolena externí knihovna Papaparse stejně jako v Metrics Graphics. Mapování jednotlivých hodnot ale už je rozdílné, jelikož tato knihovna požaduje data ve tvaru popisky, pole dat, kde proměnná popisky obsahuje pole jmen sloupců a pole dat obsahuje pole datových sad. Tyto datové sady obsahují kromě pole dat a jména sloupce, ke kterému jsou hodnoty přiřazeny, i možnosti přizpůsobení datové sady, jakožto její barvu, barvu pozadí a tloušťku ohraničení. Tyto možnosti přizpůsobení datové sady, již v rámci sady dat, zajišťuje rozpoznatelnost dat v grafu i dodatečných nastavení.

Konfigurace K vykreslení požadované vizualizace je nutné vytvořit nový objekt vizualizace Chart, ke kterému lze při jeho tvorbě definovat jeho konfiguraci. Tato konfigurace se rozděluje na dvě části. V první části se definuje typ grafu (sloupcový, spojnicový, ...) a data, která se budou vykreslovat. Tuto část je vždy povinné mít definovanou. Následuje část v rámci atributu options, která slouží pro přizpůsobení vykreslovaného grafu. Pokud tuto část uživatel nedefinuje, tak je použito základní přizpůsobení knihovny, které přidává i efekt interaktivity po najetí kurzoru myši na bod datové sady.

5.5.2 Spojnicový graf

Pro vykreslení spojnicového grafu v ChartJS je zapotřebí do atributu type vložit hodnotu "line".

Úprava typů čar v grafu Pro úpravu typů čar je nutno přímo již v rámci sady dat přidat atribut borderDash, ke kterému je nutné vložit pole hodnot, která určují, jak dlouhá má být hraniční pomlčka.

```
const dataset = {
    borderDash: [10, 1],
    ...
}
```

Listing 5.68: Příklad přidání atributu hraniční pomlčky

Nastavení baseline Pro přidání baseline bylo zapotřebí využití externího rozšíření, které nám dovolilo přidávat různé anotace do grafu. K samotné implementaci bylo ještě nutné přidat do atributu options další atribut plugin, který reprezentoval rozšíření. V tohoto nového atributu bylo následně potřeba definovat anotaci a k ní její jméno a vlastnosti.

```
plugins: {
    annotation: {
        annotations: {
          baseline: {
            type: 'line',
            mode: 'horizontal',
            scaleID: 'y',
            value: 58,
            borderColor: 'red',
            borderWidth: 1,
          }, ...
}
```

Listing 5.69: Příklad přidání baseline

Slicing Implementace slicingu je v knihovně ChartJS poměrně přímočará. V části options je pro slicing potřeba definovat atribut scales, ve kterém se následně deklaruje atribut osy, pro kterou slicing chceme provést. V tomto atributu je v poslední řadě potřeba definovat minimum a maximum, ve kterém se data mají zobrazit. Tímto způsobem je možné implementovat slicing pro zadané hodnoty.

```
scales: {
    x: {
        type: 'linear',
        min: startYear,
        max: endYear,
    }
}
```

Listing 5.70: Příklad definování slicingu

Dodatečné popisky Dodatečné popisky lze přidat stejné jako baseline za pomocí rozšíření pro annotaci. Oproti baseline se ale nastaví atribut type

na "label", definuje se v rámci atributu content, co se má vypsat a v poslední řadě je potřeba definovat pozici vyobrazení. Tato pozice se definuje za pomocí atributů xValue pro osu X a yValue pro osu Y.

```
plugins: {
    annotation: {
        annotations: {
            labelMario: {
                type: 'label',
                content: (ctx) => 'Super Mario Bros. ',
                font: {
                    size: 16
                },
                xValue: 2006,
                yAdjust: -6,
                yValue: 205.61,
            },
        ...
}
```

Listing 5.71: Příklad přidání popisku

5.5.3 Sloupcový graf

Pro vykreslení sloupcového grafu v ChartJS je zapotřebí do atributu type vložit hodnotu "bar".

Interaktivita grafů Pro implementaci interaktivity je možné v rámci atributu tooltip definovat funkci, která se provede po jeho zavolání. Tooltip je automaticky zavolán při přesunu kurzoru na daný sloupec v rámci grafu. Pro definování dalších možností interakce slouží v rámci dodatečných úprav atribut events. V tomto atributu je možné definovat požadovanou interakci a k ní přidělenou funkci.

```
tooltip: {
  callbacks: {
    label: function (context) {
      var label = datasets[context.datasetIndex].label;
      if (label) {
        label += ': ';
      }
      label += context.raw.toFixed(2) + '%';
      return label;
    }
  }
}
```

Listing 5.72: Příklad přidání popisku

Definování vlastní barvy Definovat vlastní barvu lze v knihovně za pomocí hex, rgb, rgba, hsl a hsla. Tyto barvy lze definovat v části datové sady pro atribut borderColor nebo backgroundColor.

```
borderColor: '#FF6384',
backgroundColor: 'rgb(54, 162, 235)',
```

Listing 5.73: Příklad definování vlastních barev

Škálovatelnost os Pro nastavení škálovatelnosti os je zapotřebí v atributu scales pro danou osu přidat atribut type, čímž určíme typ škálování dané osy.

```
scales: {
    y: {
        type: 'logarithmic',
    }
...
}
```

Listing 5.74: Příklad definice škálování

Stacked a stacked 100% Pro vykreslení grafu v rámci stacked verze je pouze potřeba nastavit typ škálování na kategorické a atribut stacked na true a graf bude vyobrazen ve stacked verzi.

```
scales: {
    x: {
        type: 'category',
        stacked: true
    }
    y: {
      stacked: true
    }
...
}
```

Listing 5.75: Příklad definice škálování

5.5.4 Scatter plot

Pro vykreslení scatter plot grafu v ChartJS je zapotřebí do atributu type vložit hodnotu "scatter".

Modifikace dat Modifikovat data lze pouze v datové sadě, kde je možné přidat dodatečné atributy jako pointRadius, pointStyle a pointRadius a další. Tyto atributy umožňují úpravu stylu dat, kterou uživatel požadoval.

```
data: {
    datasets: [{
        ...
        backgroundColor: 'rgba(54, 162, 235, 0.5)',
        pointRadius: 5,
        pointStyle: 'rectRot',
        pointRadius: 5
        ...
    }]
},
```

Listing 5.76: Příklad modifikace dat

Posunutí báze Pro posunutí báze v knihovně ChartJS je potřeba v atributu scales definovat pro požadovanou osu atribut Min. Tento atribut způsobí, že data budou vykreslena pouze od tohoto minima, čímž se posune osa grafu na požadovanou hodnotu

```
scales: {
    y: {
        Min: 50,
        ...
    }
}
```

Listing 5.77: Posunutí báze osy Y na hodnotu 50

Filtrace dat Filtrace dat lze dosáhnout za pomocí využití JavaScriptové funkce filter ("podmínka") na sadu dat. V této funkci je potřeba definovat podmínku, kterou mají data splňovat. Následně jsou vyfiltrovaná data použita ve vizualizaci.

```
var filteredData = data.filter(entry => parseFloat(entry.
    ↪ Critic_Score) >= 60);
```

Listing 5.78: Příklad filtrace

Modifikace legendy Modifikovat legendu je možné za pomocí atributu plugin v dodatečné konfiguraci. V tomto atributu je poté nutné deklarovat atribut legend a pro něj všechny jeho požadované modifikace.

```
plugins:{
    legend: {
        display: true,
        position: 'right',
        labels: {
            fontColor: 'black',
            fontStyle: 'bold',
            boxWidth: 20
        }
    }
}
```

Listing 5.79: Příklad modifikace legendy

5.5.5 Paprskový graf

Pro vykreslení paprskového grafu v ChartJS je zapotřebí do atributu type vložit hodnotu "radar".

Popisky os V knihovně je poměrně jednoduché vkládat popisky k jednotlivým osám. To se provádí v konfiguraci jednotlivých os za pomocí atributu title.

```
y: {
    ...
    title: {
        display: true,
        text: 'Critic Score'
    }
}
```

Listing 5.80: Příklad přidání popisků

Výplň datové sady Pro nastavení výplně jednotlivých vizualizací je potřeba v datové sadě nastavit atribut backgroundColor na požadovanou barvu výplně

```
const datasets ={
    ...
    backgroundColor: backgroundColor,
    ...
}
```

Listing 5.81: Příklad přidání výplně

5.6 Zhodnocení

Jednotlivé vizuální provedení grafů v uvedených knihovnách lze nalézt v příloze.

Metrics Graphics V knihovně Metrics Graphics je poměrně jednoduché vizualizovat základní typy grafů jako jsou spojnicové, scatter plot a sloupcové grafy. Konfigurace takovýchto grafů je poměrně intuitivní bez nutnosti definovat různé obalovací množiny konfigurací. Ohledně možností přizpůsobení jednotlivých vizualizací knihovna částečně pokulhává. Ve zkoušené verzi se objevily problémy jako nefunkční generické přizpůsobení os ve sloupcovém grafu, což znesnadnilo práci s danou knihovnou. Kromě tohoto problému se nic dalšího nevyskytlo. Jako nedostatek této knihovny by se dala považovat podpora pouze menší množiny přizpůsobení pro jednotlivé osy, data, linie a popisky. Jedná se tedy o knihovnu, která splňuje základní možnosti vizualizace a nabízí některá specifická přizpůsobení, která umožňují si graf upravit, aby alespoň částečně odpovídal našim požadavkům.

V knihovně byly implementovány funkcionality:

- Nastavení baseline
- Popisky os
- Markery
- Výplň datové sady
- Interaktivita grafů
- Definování vlastní barvy
- Škálovatelnost os
- Modifikace dat
- Posunutí báze

Google Charts Knihovna Google Charts umožňuje jednoduché a velmi přehledné vizuální zpracování pro všechny základní typy podporovaných typů grafů. Konfigurace těchto grafů je oddělena do samostatného objektu, což je velmi dobrý způsob, jak využít různé přizpůsobení v odlišných cyklech života grafu. Knihovna obsahuje proti Metrics Graphics více možností, jak přizpůsobit vybranou vizualizaci, a soustředí se na vykreslení všech možných typů dat, kdežto Metrics Graphics se soustředí z velké části jen na datové řady. V rámci implementace se neobjevily větší problémy, které by zabraňovaly plynulé konfiguraci a používání knihovny. Souhrnem se tedy jedná o knihovnu, která nabízí dobrý poměr přizpůsobení/složitosti pro jednoduché i částečně pokročilé grafy. Tato přizpůsobení umožnila upravit graf do požadovaného vizuálu.

V knihovně byly implementovány funkcionality:

- Úprava typů čar v grafu
- Nastavení baseline
- Slicing
- Dodatečné popisky
- Interaktivita grafů
- Definování vlastní barvy
- Škálovatelnost os
- Stacked a stacked 100
- Modifikace dat
- Posunutí báze
- Filtrace dat
- Stylizování os grafu
- Popisky os
- Markery
- Modifikace legendy
- Předdefinované sady barev

Vega Knihovna Vega, Vega-lite a Vega-embed společně umožňují vytvářet jak jednoduché, tak i komplexní grafy za pomocí JSON formátu. Díky tomu, že tato knihovna podporuje i složitější typy grafů, je i konfigurace takovýchto grafů komplikovanější. Naštěstí knihovna stále obsahuje atributy, které oddělují jednotlivé části grafu, a tím pádem je jejich konfigurace poměrně přehledná a jasně definovaná. Při práci s touto knihovnou se objevil pouze jeden menši problém, kdy přiřazení barvy nefungovalo podle uvedené podmínky, což bylo nutné obejít jinou formou výpočtu barvy. Mimo tento problém byla práce s knihovnou v rámci složitosti poměrně odpovídající komplexitě grafu. Provedená přizpůsobení umožnila upravit graf do požadovaného vizuálu.

V knihovně byly implementovány funkcionality:

- Úprava typů čar v grafu
- Nastavení baseline
- Slicing
- Dodatečné popisky
- Interaktivita grafů
- Definování vlastní barvy
- Škálovatelnost os
- Stacked a stacked 100
- Modifikace dat
- Posunutí báze
- Filtrace dat
- Stylizování os grafu
- Popisky os
- Výplň datové sady
- Markery
- Modifikace legendy

D3 Tato knihovna obdobně jako Vega umožňuje vytváření jednoduchých, ale i komplexních grafů. Tato knihovna má z analyzované množiny zcela největší komplexnost a její konfigurace je taktéž nejkomplikovanější. Díky jednotlivým objektům pro vykreslované prvky je sice možné je přizpůsobit skoro libovolným způsobem, ale zakomponování několika těchto objektů v rámci komplexnějšího typu grafu je velmi složité. Při práci s touto knihovnou bylo nejsložitější vypočítání jednotlivých pozic pro právě přidávané prvky. Kromě problému se složitostí knihovny se jedná o knihovnu, která umožňuje uživateli přizpůsobit skoro cokoliv, na co si vzpomene.

V knihovně byly implementovány funkcionality:

- Úprava typů čar v grafu
- Nastavení baseline
- Slicing
- Dodatečné popisky
- Interaktivita grafů
- Definování vlastní barvy
- Škálovatelnost os
- Stacked a stacked 100
- Modifikace dat
- Posunutí báze
- Filtrace dat
- Stylizování os grafu
- Popisky os
- Výplň datové sady
- Markery
- Modifikace legendy
- Předdefinované sady barev

ChartJS ChartJS je v rámci vykreslování grafů někde na pomezí Metrics Graphics a Google Charts. ChartJS oproti těmto knihovnám poskytuje možnost vizualizace i několika pokročilých grafům. Konfiguraci grafů v ChartJS je možné oddělit od samotného grafu jako v knihovně Google Charts. Knihovna v základu obsahuje více možností přizpůsobení než Metrics Graphics, ale méně než Google Charts. Pokud ale jsou přidána dodatečná rozšíření, tak tato knihovna umožňuje přizpůsobení na pomezí Google Charts a Vega. Při práci s knihovnou se nevyskytly žádné komplikace. Tato knihovna se řadí k těm méně složitým a byla by schopná jednoduše zastoupit například Metrics Graphics.

V knihovně byly implementovány funkcionality:

- Úprava typů čar v grafu
- Nastavení baseline
- Slicing
- Dodatečné popisky
- Interaktivita grafů
- Definování vlastní barvy
- Škálovatelnost os
- Stacked a stacked 100
- Modifikace dat
- Posunutí báze
- Filtrace dat
- Stylizování os grafu
- Popisky os
- Výplň datové sady
- Markery
- Modifikace legendy

Funkcionality	Metrics Graphics	Google Charts	Vega	D3	ChartJS
Interaktivita grafů	3	3	4	5	3
Definování vlastní barvy	1	1	1	1	1
Škálovatelnost os	2	2	3	4	2
Úprava typů čar v grafu	N/A	1	3	3	1
Nastavení baseline	1	3	3	3	1
Slicing	N/A	2	3	4	2
Dodatečné popisky	N/A	2	3	4	2
Modifikace dat	3	2	4	4	1
Posunutí báze	2	2	3	3	2
Filtrace dat	N/A	3	3	3	3
Stylizování os grafu	N/A	2	3	3	2
Popisky os	2	2	3	4	2
Výplň datové sady	2	N/A	2	4	1
Markery	1	4	4	4	1
Modifikace legendy	N/A	2	3	3	2
Předdefinované sady barev	N/A	1	N/A	1	N/A

Obrázek 5.1: Tabulka implementovaných funkcionalit a jejich obtížnosti

V tabulce 5.1 je uvedený souhrn všech implementovaných funkcionalit společně, kde 5 znamená těžkou implementaci a 1 lehkou.

6 Závěr

V rámci této bakalářské práce byly popsány základní koncepty efektivního přístupu k vizualizaci grafů. Následně byla sestavena množina funkčních požadavků pro přizpůsobení vizualizací společně s množinou grafů, které byly zkoušeny v implementační části. Po definování těchto množin byla vytvořena množina ze zajímavých JavaScriptových knihoven, ve kterých byly dané funkcionality a grafy zkoušeny. V rámci této množiny byly jednotlivé knihovny představeny a bylo zde i poukázáno na to, jaké technologie daná knihovna využívá.

Poté následovala implementační část, kde v rámci každé knihovny byly implementovány všechny funkcionality z definované množiny, které daná knihovna podporovala. V rámci této kapitoly byla detailněji rozebrána i datová část jednotlivých knihoven, kde bylo poukázáno na to, jak jednotlivé knihovny zpracovávají předaná data a jestli poskytují nějaké funkce, které jsou schopny ulehčit načítání CSV souborů, a nebo jestli je nutné pro danou knihovnu využít vlastní skript nebo externí knihovnu.

V poslední kapitole byly shrnuty jednotlivé knihovny. V tomto shrnutí byly obsaženy informace, jak složité bylo implementovat množinu funkcionalit, jaké problémy se při implementaci vyskytly, jaké bylo rozvržení jednotlivých konfiguračních souborů, jak splnila daná knihovna očekávání, zajímavé poznatky z knihovny a zhodnocení, jestli složitost knihovny je relevantní k její možnosti přizpůsobení. Z tohoto zhodnocení vyšlo najevo, že Metrics Graphics, Google Charts a ChartJS jsou poměrně jednoduché knihovny na přizpůsobení, ale některé z nich neumožňují veškeré možnosti přizpůsobení. Z těchto knihoven vyšlo také najevo, že pokud uživatel chce jednoduché přizpůsobení grafu, tak je nejlepší vybrat si knihovnu Google Charts nebo ChartJS. Pokud uživatel potřebuje složitější přizpůsobení nebo nestandardní grafy, tak je v podstatě jedno, jestli využije knihovnu D3 nebo Vega, i když Vega obsahuje oproti D3 strukturovanou konfiguraci.

Jednotlivé informace z bakalářské práce jsou úzce propojeny s konkrétními verzemi použitých knihoven. Pokud by čtenář měl zájem hodnotit jiné verze těchto knihoven, bude nutné provést novou analýzu. V různých verzích knihoven může docházet k změnám v podpoře jednotlivých funkcionalit. To znamená, že některé funkcionality mohou být ve vyšších verzích knihoven odstraněny, zatímco v jiných verzích mohou být přidány funkcionality nové.

Příloha A - Seznam zkratek

OS - Operating system

AJAX - Asynchronous JavaScript and XML

CSV - Comma-separated values

SVG - Scalable Vector Graphics

HTML - HyperText Markup Language

API - Application Programming Interface

JSON - JavaScript Object Notation

CSS - Cascading Style Sheets

DOM - Document Object Model

XML - Extensible Markup Language

Literatura

[1] KNAFLIC, C. N. *Storytelling with Data: A Data Visualization Guide for Business Professionals*. John Wiley Sons, 2015. ISBN 978-1-119-00225-3.

[2] TUFTE, E. *The Visual Display of Quantitative Information*. Graphics Press, 1983. ISBN 9780961392109.

[3] YI, M. *How to Choose Colors for Data Visualizations* [online]. Chartio, 2021. [cit. 2023/20/05]. Dostupné z: `https://chartio.com/learn/charts/how-to-choose-colors-data-visualization`.

[4] *DataViz Cheatsheet* [online]. Policyviz, 2018. [cit. 2023/20/05]. Dostupné z: `https://policyviz.com/2018/08/07/dataviz-cheatsheet/`.

[5] MORRIS, A. *What Is a Chart Why Is It Important for Businesses?* [online]. Oracle, 2021. [cit. 2023/20/05]. Dostupné z: `https://www.netsuite.com/portal/resource/articles/erp/chart.shtml`.

[6] *From Data to Viz* [online]. DatatoViz, 2018. [cit. 2023/20/05]. Dostupné z: `https://www.data-to-viz.com/#heatmap`.

[7] GREGORY, S. *Video Game Sales* [online]. Kaggle, 2016. [cit. 2023/20/05]. Dostupné z: `https://www.kaggle.com/datasets/gregorut/videogamesales`.

[8] XTYSCUT. *Video games sales as at 22 Dec 2016* [online]. Kaggle, 2017. [cit. 2023/20/05]. Dostupné z: `https://www.kaggle.com/datasets/xtyscut/video-games-sales-as-at-22-dec-2016csv`.

[9] *JavaScript* [online]. Mozilla, 2023. [cit. 2023/20/05]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/JavaScript`.

[10] *JavaScript History* [online]. W3schools, 2023. [cit. 2023/20/05]. Dostupné z: `https://www.w3schools.com/js/js_history.asp`.

[11] *About* [online]. Metrics Graphics, 2023. [cit. 2023/20/05]. Dostupné z: `https://metricsgraphicsjs.org/`.

[12] *Overview* [online]. Google, 2022. [cit. 2023/20/05]. Dostupné z: `https://developers.google.com/chart/interactive/docs`.

[13] *Vega – A Visualization Grammar* [online]. Vega, 2023. [cit. 2023/20/05]. Dostupné z: `https://vega.github.io/vega/`.

[14] *What is D3?* [online]. Mike Bostock and Observable, 2023. [cit. 2023/20/05]. Dostupné z: `https://d3js.org/`.

[15] *Why Chart.js* [online]. Chart.js, 2023. [cit. 2023/20/05]. Dostupné z: `https://www.chartjs.org/docs/4.3.0/`.

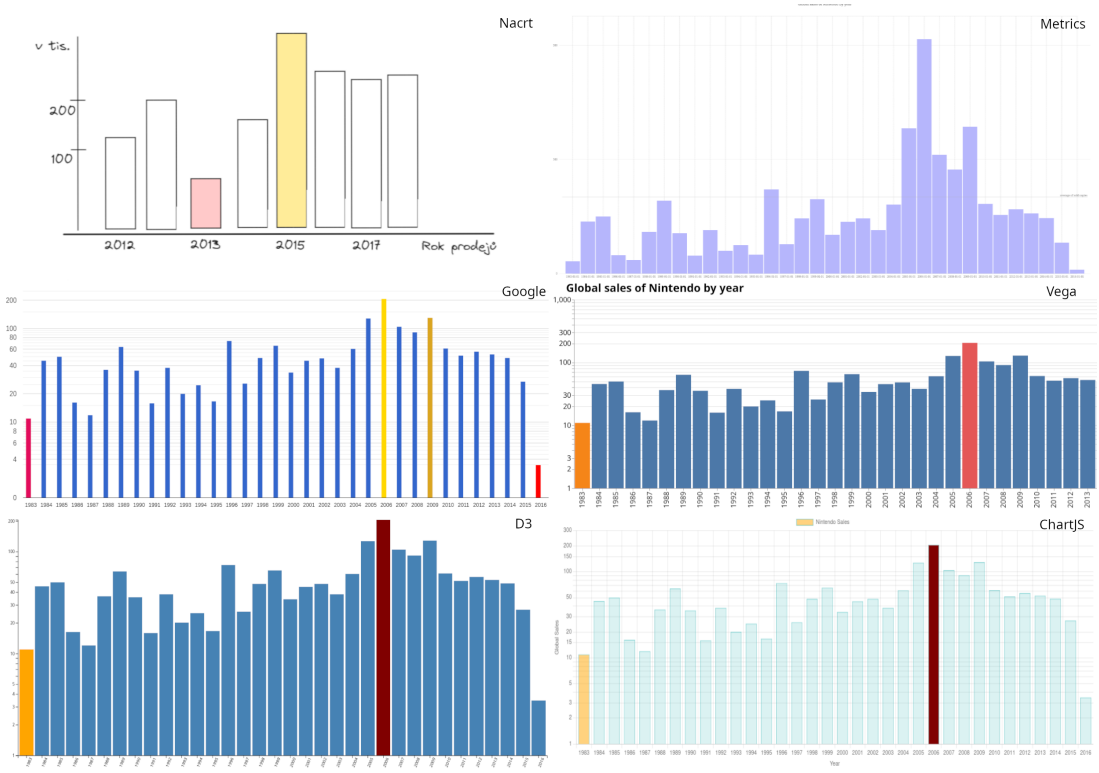
Příloha B - Elektronická příloha

Adresář A20B0596P_prilohy obsahuje 4 podadresáře.

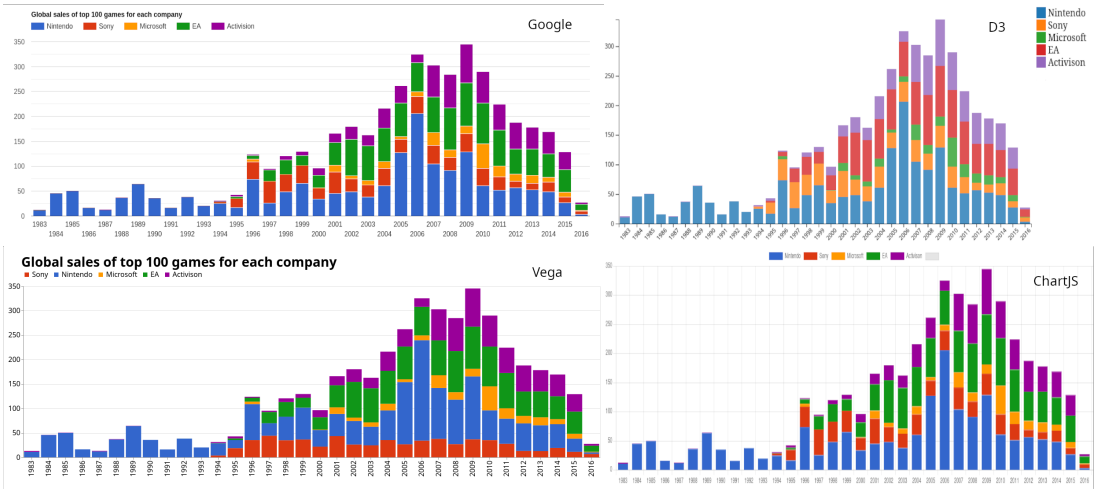
1. **A20B0596P_prilohy/Text_prace** - obsahuje PDF soubor s textem bakalářské práce, adresář s TeX zdrojovým dokumentem a adresář s obrázky použitými v textu.
2. **A20B0596P_prilohy/Aplikace_a_knihovny** - obsahuje jednotlivé kódy implementovaných grafů napříč všemi knihovnami.
3. **A20B0596P_prilohy/Vstupni_data** - obsahuje datové soubory použité v příkladech.
4. **A20B0596P_prilohy/Vysledky** - obsahuje výsledné vizualizace

Každý z těchto adresářů obsahuje také Readme.txt soubor s detailnějšími informacemi.

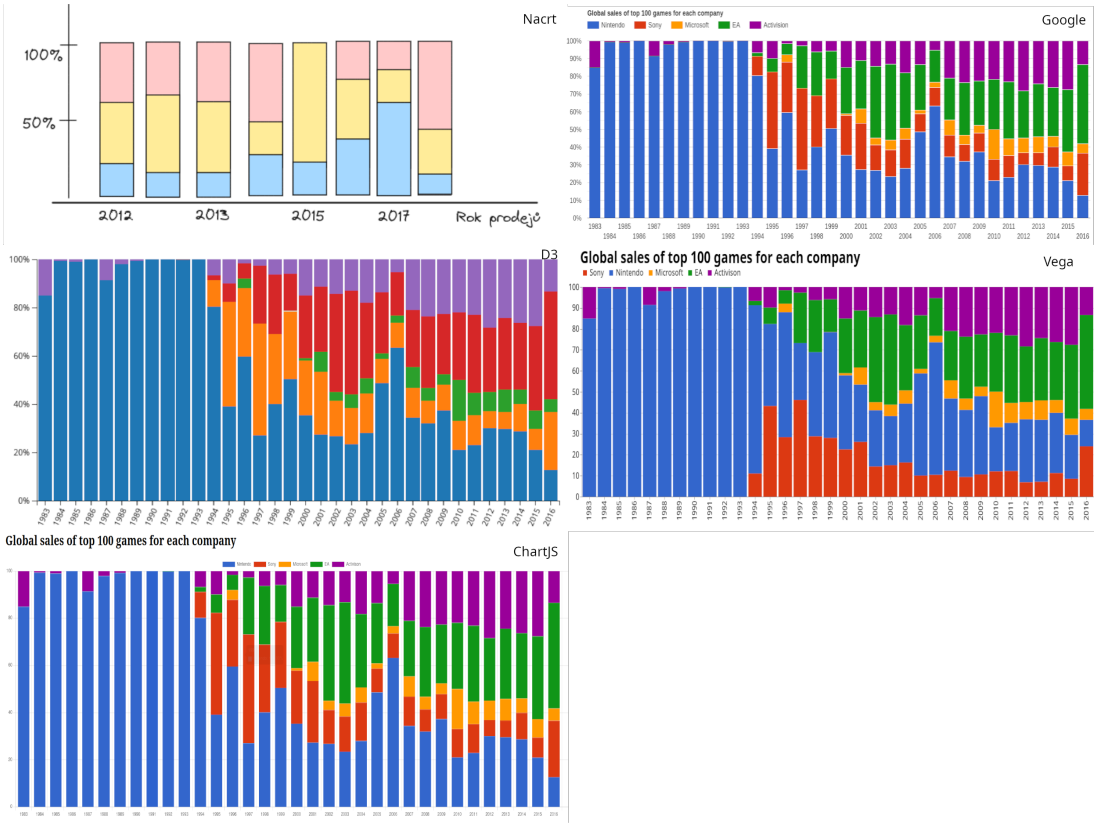
Příloha C - Vytvořené vizualizace



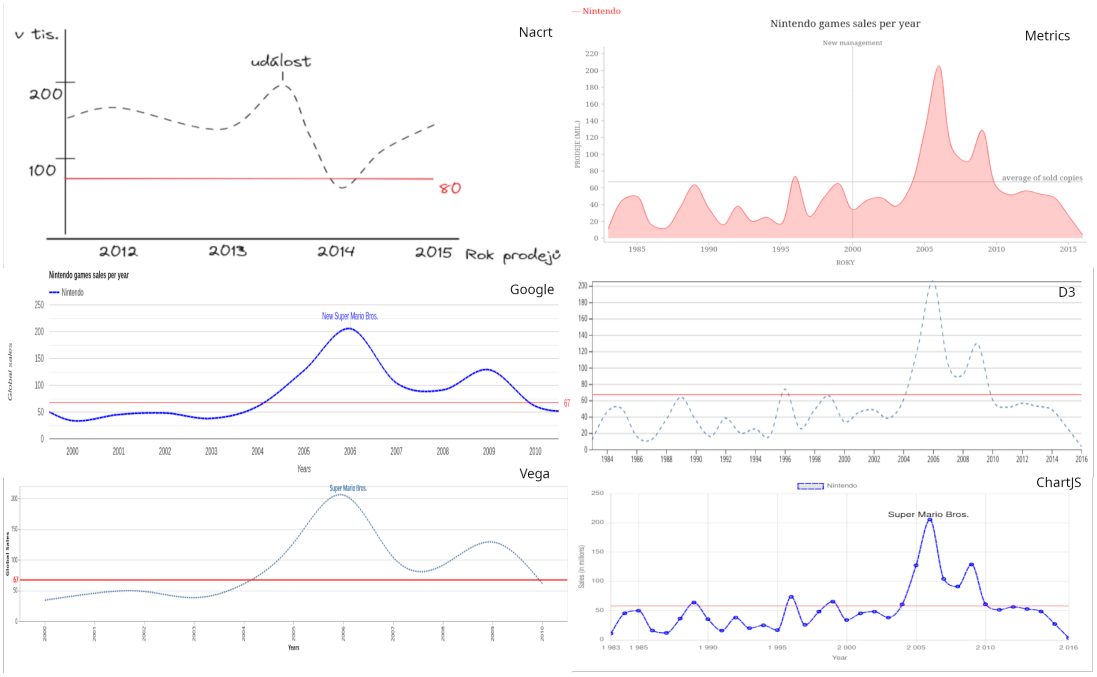
Obrázek 6.1: Sloupcové grafy s jednou kategorií



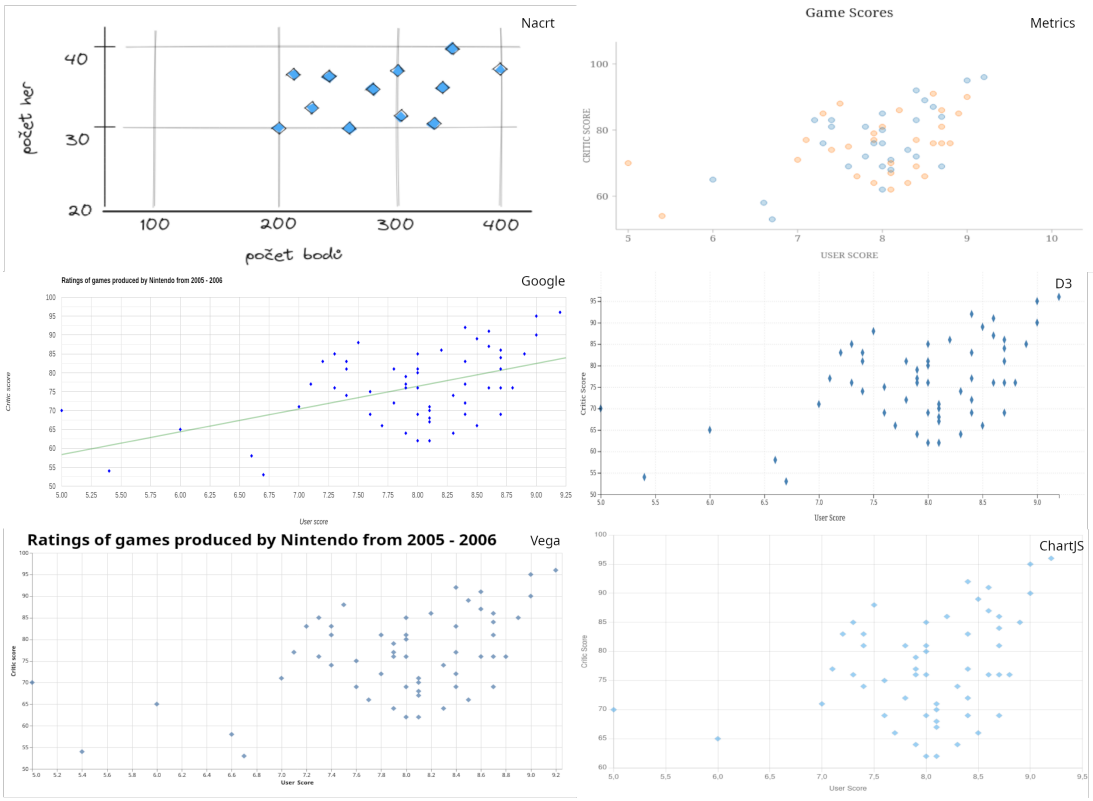
Obrázek 6.2: Sloupcové grafy s více kategoriemi



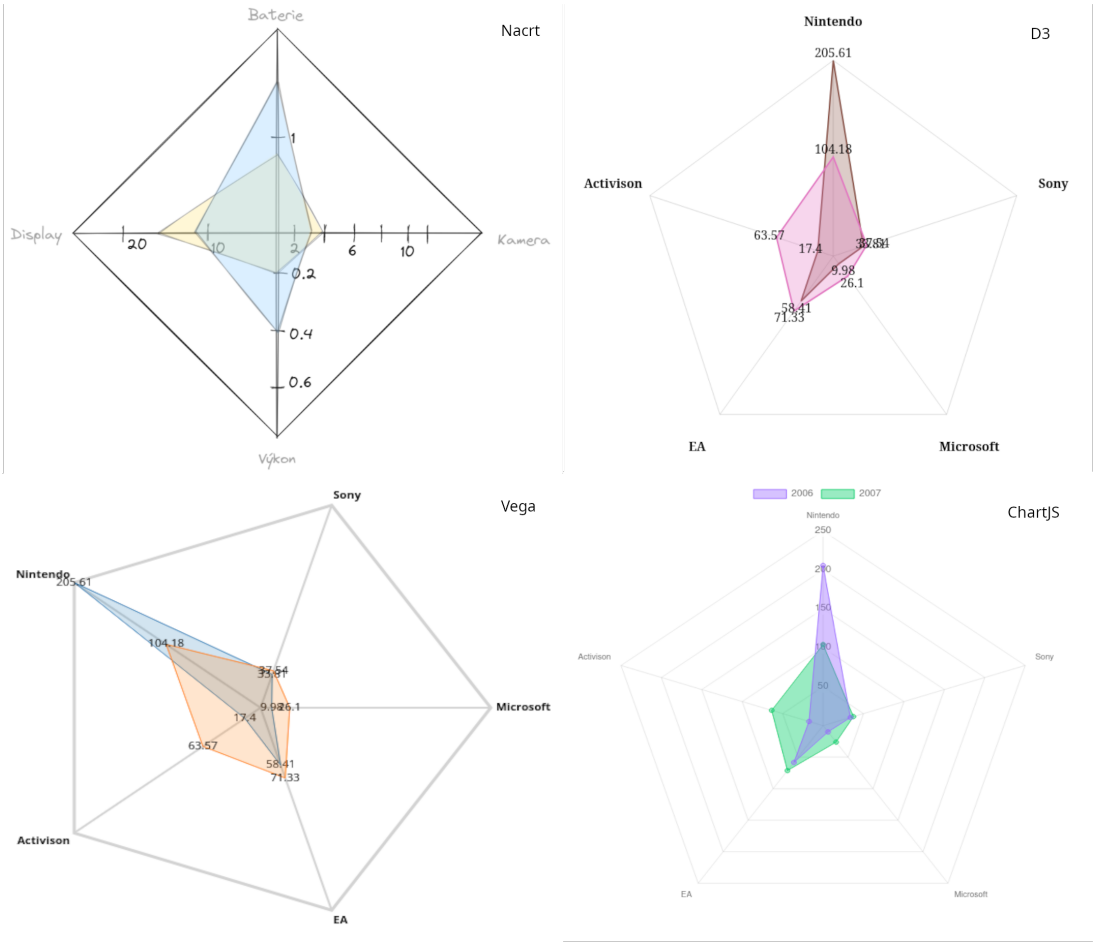
Obrázek 6.3: 100% Sloupcové grafy s více kategoriemi



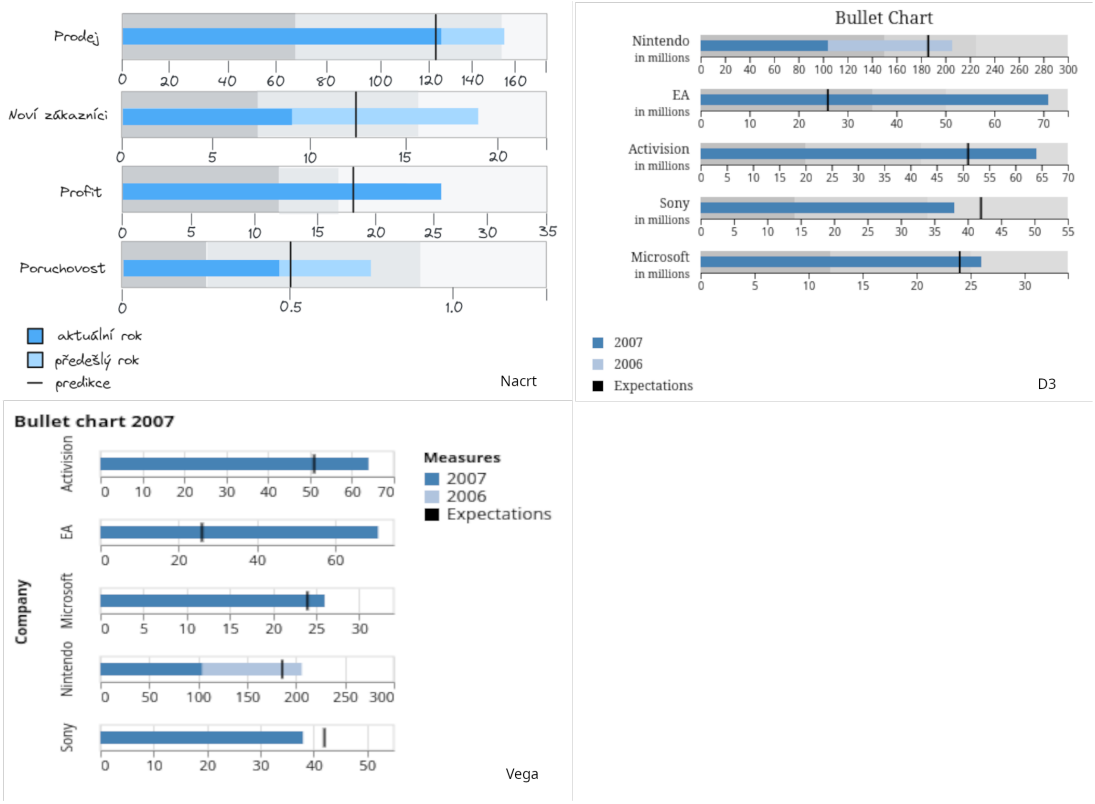
Obrázek 6.4: Spojnicové grafy



Obrázek 6.5: Bodové grafy



Obrázek 6.6: Paprskové grafy



Obrázek 6.7: Bullet grafy