FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

# Zadání bakalářské práce

| | |
|---|---|
| **Název:** | Webová aplikace pro tvorbu interaktivních grafů |
| **Student:** | František Špaček |
| **Vedoucí:** | Ing. Marek Suchánek, Ph.D. |
| **Studijní program:** | Informatika |
| **Obor / specializace:** | Webové inženýrství 2021 |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | do konce letního semestru 2024/2025 |

## Pokyny pro vypracování

Řada webových stránek, aplikací a portálů využívá externích služeb pro vykreslování a vkládání interaktivních grafů. Bohužel však tyto služby často nedisponují dostatečnými možnostmi konfigurace, nelze je přizpůsobit vhodně požadovanému vzhledu, jsou náročné na výkon, nebo mají jiné další nevýhody zabraňující snadnému použití. Cílem této práce je vyvinout nové řešení, které redukuje tyto nedostatky a poskytne praktickou alternativu, kterou bude možné dále jednoduše rozšiřovat. V rámci práce bude postupováno s metodami softwarového inženýrství:

1. Analyzujte problematiku využití grafů pro vizualizaci dat (různé typy grafů, společné vlastnosti a možnosti). Dále stručně popište již existující způsoby řešení vykreslování grafů ve webových aplikacích.
2. Proveďte stručnou rešerši nabízených služeb poskytujících tvorbu interaktivních grafů a jejich vkládání do jiných aplikací, webů a portálů. Shrňte klíčové vlastnosti a nedostatky těchto řešení.
3. Sestavte požadavky a případy užití pro vlastní řešení.
4. Navrhněte vlastní řešení s ohledem na splnění požadavků (dle priorit), rozšiřitelnost a využitelnost. Výběr technologií řádně zdůvodněte.
5. Implementujte, otestujte a zdokumentujte prototyp řešení dle návrhu.
6. Zhodnoťte přínosy vlastního řešení v porovnání s existujícími a navrhněte další možný rozvoj aplikace.

---

*Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 16. listopadu 2023 v Praze.*

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

# Webová aplikace pro tvorbu interaktivních grafů

***František Špaček***

Katedra softwarového inženýrství
Vedoucí práce: Ing. Marek Suchánek, Ph.D.

16. května 2024

# Poděkování

Děkuji Ing. Markovi Suchánkovi Ph.D. et Ph.D. za vedení práce. Dále bych rád poděkoval Ing. Jiřímu Jirkovcovi a Ing. Ivě Špačkové za trpělivost a podporu při studiu.

# Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 16. května 2024

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2024 František Špaček. Všechna práva vyhrazena.
*Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.*

### Odkaz na tuto práci

Špaček, František. *Webová aplikace pro tvorbu interaktivních grafů*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2024. Dostupný také z WWW: ⟨`https://spacek.blue`⟩.

# Abstrakt

Tato bakalářská práce se zabývá interaktivními grafy ve webovém prostředí. Nejprve se zaměřuje na analýzu již existujících řešení a na porovnání různých přístupů. Na základě průzkumu jsou sestaveny požadavky na aplikaci a případy užití. Následuje návrh a implementace řešení. Na závěr je aplikace otestována, nasazena na server a porovnána s ostatními řešeními. Výsledkem je webová aplikace umožňující tvorbu interaktivních grafů, které lze následně vkládat na další webové stránky.

**Klíčová slova** webová aplikace, grafy, knihovna

# Abstract

This bachelors thesis is about interactive charts in web environment. It first focuses on analysis of existing solutions and compares different approaches. It compiles functional and non functional requirements and use cases based on the previous analysis. Next is the designing and implementation of the application. Finally, the app is deployed and tested against existing solutions. The result is a web application that enables the creation of interactive charts that can then be inserted into other web pages.

**Keywords** web application, charts, library

# Obsah

# Seznam obrázků

# Seznam zdrojových kódů

# Úvod

V dnešní digitální éře se stále více spoléháme na vizualizaci dat k tomu, abychom lépe porozuměli složitým vzorcům, trendům a vztahům. Řada webových stránek, aplikací a portálů využívá externích služeb pro vykreslování a vkládání interaktivních grafů. Bohužel však tyto služby často nedisponují dostatečnými možnostmi konfigurace, nelze je přizpůsobit vhodně požadovanému vzhledu nebo mají jiné další nevýhody zabraňující snadnému použití. Uživatel si může mezi jednotlivými službami vybírat. Pokud chce více grafů či veliké možnosti úprav, tak určitě nějakou službu nalezne. Všechny však mají jeden společný problém, a tím jsou zbytečně vysoké nároky na výkon.

Tato práce se snaží přinést nové řešení, které nabízí všechny výhody existujících služeb a zároveň má nižší požadavky na hardwarové prostředky a tím pádem funguje plynule i na slabších zařízeních.

Úvodní část práce zahrnuje analýzu současného stavu problematiky a existujících řešení, identifikaci požadavků a cílů aplikace a návrh její architektury a funkčnosti. Dále se práce zabývá implementací navrženého systému včetně vývoje uživatelského rozhraní, backendové logiky a databáze.

Další část práce je zaměřena na testování vytvořené aplikace, které zahrnuje ověření správnosti funkcionality, uživatelského prostředí a výkonu systému.

V závěrečné části je nové řešení nasazeno do serverového prostředí, což zahrnuje konfiguraci, zprovoznění aplikace a zajištění bezpečnosti a dostupnosti pro uživatele.

Výsledkem této práce je nové řešení, jež redukuje existující nedostatky a poskytuje praktickou alternativu, kterou bude možné dále jednoduše rozšiřovat. Tento projekt si klade za cíl vytvořit takový nástroj, který umožní uživatelům vytvářet, upravovat a vizualizovat grafy na základě jejich dat bez zbytečných omezení. Vytvořené grafy budou rychlé a přívětivé i pro slabší zařízení.

Kapitola 1

# Cíl práce

Cílem této bakalářské práce je vytvořit knihovnu pro vykreslování interaktivních grafů ve webovém prostředí a nabídnout podpůrnou aplikaci pro tvorbu těchto grafů. Projekt si klade za cíl vyvinout nové řešení pro vytváření, upravování a sdílení interaktivních grafů, které bude poskytovat uživatelům větší flexibilitu, snadnou konfiguraci a zároveň efektivní využití výpočetních zdrojů.

Dílčími cíli je provést detailní analýzu existujících služeb a technologií v oblasti tvorby a vykreslování interaktivních grafů, aby bylo možné identifikovat klíčové potřeby uživatelů a zajistit adekvátní funkčnost nového řešení. Následuje návrh a implementace knihovny a aplikace. Nakonec přichází na řadu důkladné testování a ověření funkčnosti a uživatelské přívětivosti vytvořené aplikace.

Kapitola 2

# Analýza

Jako první krok v rámci tohoto projektu je vhodné provést analýzu již existujících služeb. Cílem této kapitoly je poskytnout přehled o tom, jaké funkce a možnosti tyto služby nabízejí, jakým způsobem fungují, a jak se v praxi osvědčují z uživatelského hlediska. Vedle toho je důležité zkoumat různé přístupy a technologie, které se na trhu používají při vytváření grafů, a identifikovat jejich silné a slabé stránky.

Na základě sesbíraných dat jsou sestaveny funkční i nefunkční požadavky a případy užití, které pomáhají určit jakým směrem se má vývoj řešení vydat, aby výsledný produkt byl efektivní a odpovídal uživatelským potřebám.

## 2.1 Podobné aplikace

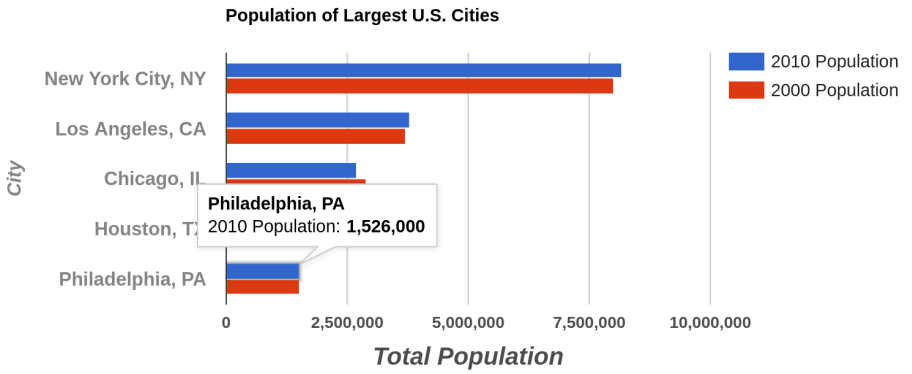
### 2.1.1 Google charts

Charts[1] od firmy Google je v současné chvíli nejvíce používanou službou pro vykreslování grafů na trhu. Kromě rozsáhlého výběru grafů nabízí také podrobnou a přehled dokumentaci, která uživatelům usnadňuje rychle se zorientovat ve všech dostupných funkcích a v tom, jak se s Charts má interagovat. Služba je bezplatná, a přesto zahrnuje i pokročilé funkce. Určena je pro všechny druhy projektů, od školní výuky a tvorby osobních stránek až po rozsáhlé aplikace.

#### 2.1.1.1 Prostředí

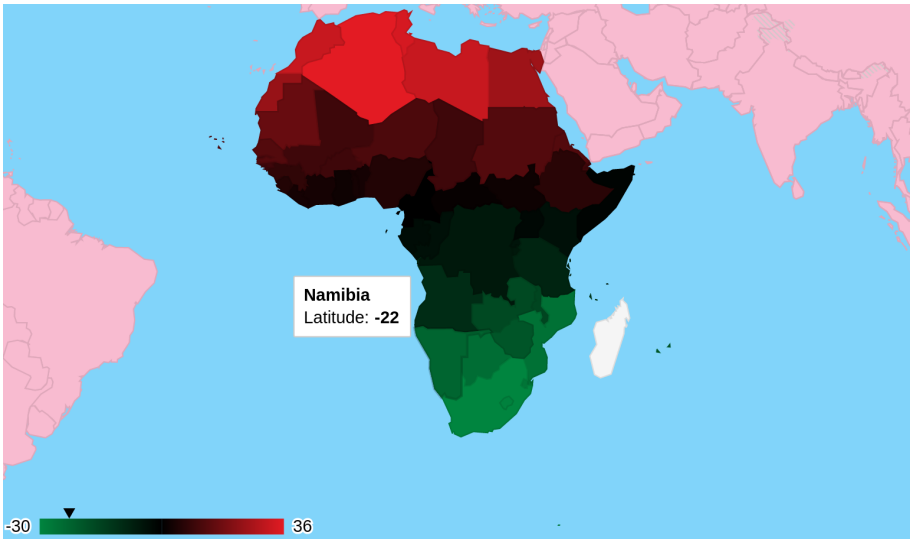
Uživatelé si mohou vybírat z desítek různých typů grafů, od tradičních spojnicových a sloupcových grafů až po méně obvyklé formáty včetně bublinových a paprskových grafů. Tato rozmanitost umožňuje uživatelům vybrat ten nejvhodnější typ grafu pro konkrétní potřeby a data. Jedním z hlavních přínosů Google Charts je možnost rozsáhlého přizpůsobení grafů. Uživatelé mohou na příklad měnit barvy, nastavovat osy, legendu či efekty a upravovat další vizuální aspekty grafů podle požadovaných preferencí. Všechny grafy jsou v Google Charts prezentovány ve 2D formátu, což zajišťuje jednoduchost a srozumitelnost prezentovaných dat.

Kromě běžných grafů, jako je vidět na obrázku 2.1, jsou také k dispozici exotičtější způsoby zobrazení dat. Mezi takové patří například histogramy, Gantův diagram, hierarchické stromy či vykreslování dat na mapu. Pokud tento




Obrázek 2.1: Google horizontální sloupcový graf [1]

základní výběr nestačí, tak je lze případě potřeby také spojit více grafů dohromady, čímž se násobně zvýší počet dostupných možností.




Obrázek 2.2: Google GeoChart [1]

Jednou z klíčových vlastností Google Charts je jeho interaktivita. Uživatelé mohou snadno zvýraznit jednotlivé hodnoty v grafu, což umožňuje rychlou identifikaci klíčových datových bodů a trendů. Dále mohou zobrazit podrobnější informace o datech přímo v grafu, což poskytuje hlubší pohled do dat a umožňuje lepší porozumění prezentovaným informacím. Případně si je může vyznačit v legendě, která může být volitelně zobrazena.

#### 2.1.1.2 Zprovoznění

Přestože Charts vyniká ve finálním vykreslení grafu a v přívětivosti pro koncového uživatele, nedá se to samé říct o implementaci na webové stránky.

Nejprve je zapotřebí naimportovat JavaScriptové knihovny které se starají o samotné vykreslování. Následně je nutné vytvořit div, do kterého se graf

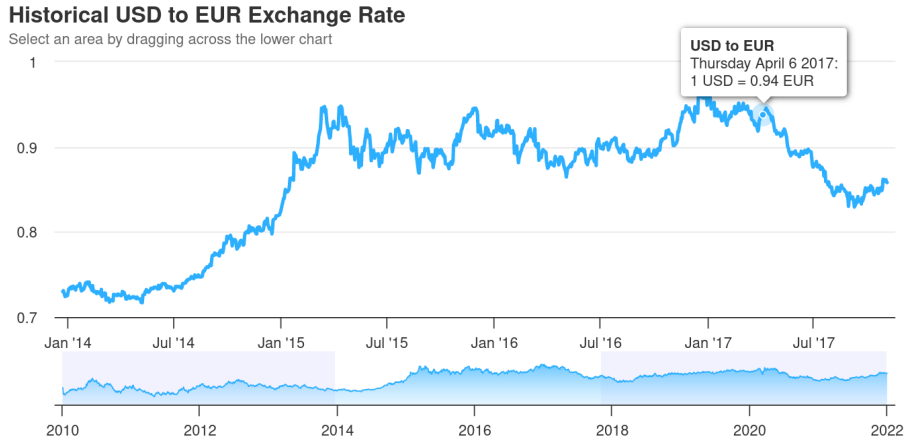
vykreslí pomocí SVG[1]. Všechny možnosti grafu se nastavují pomocí scriptu, což je velice nepřehledné a existuje zde proto mnoho možností pro chybu. Existují dvě možnosti vkládání dat. Pro první možnost musíme všechna data, se kterými chceme pracovat, natvrdo zapsat do kódu. Výhodou je, že není potřeba žádná externí služba a všechno se odehrává pouze na úrovni dané stránky, zároveň se tím však velice komplikují případné budoucí změny.

### 2.1.2 Highcharts

HighCharts[2] se specializuje spíše na odborné, a tedy více podrobné grafy s velikým množstvím zobrazovaných dat. Jedná se o komerční službu, která nabízí i bezplatnou verzi, kde jsou však uživateli přístupné pouze základní funkce. Pokud chce uživatel využívat všech dostupných funkcí, musí si zaplatit licenci, které začínají na částce čtyři tisíce korun za rok.

#### 2.1.2.1 Prostředí

HighCharts nabízí širokou škálu grafických možností zahrnující přibližně stovku různých typů grafů. Tento rozsáhlý výběr zahrnuje bodové, bublinové a paprskové grafy včetně jejich trojrozměrných variant. Jsou schopny zobrazovat velké množství dat současně, a to včetně různých typů popisků a detailních informací, což je činí ideálním nástrojem pro prezentaci komplexních datových sad. Kromě grafů jsou k dispozici i další možnosti zobrazení dat, které mohou být využity pro různé účely a potřeby uživatelů.




Obrázek 2.3: Highcharts Master-detail graf [2]

#### 2.1.2.2 Zprovoznění

Implementace HighCharts je velice komplikovaná a běžný uživatel by si s ní nevěděl rady. Pro uvedení do provozu je zapotřebí velikého množství knihoven a dalších služeb. Tato složitost je dána i tím, že není služba vázána pouze na jednu platformu nebo jeden programovací jazyk, mezi něž patří například JavaScript, Python, .NET, R a další.

[1] Scalable Vector Graphics

I když může být proces nastavení z počátku složitý, tak HighCharts nabízí komplexní a intuitivní uživatelské rozhraní pro správu a úpravu grafů po jejich nasazení. To znamená, že po úvodní instalaci a konfiguraci není nutné stále zasahovat do kódu. Uživatelé mohou snadno provádět změny, aktualizace a přizpůsobení přímo z integrovaného uživatelského prostředí HighCharts. Data je možné vkládat přímo, nebo automaticky po připojení k databázi.[9]

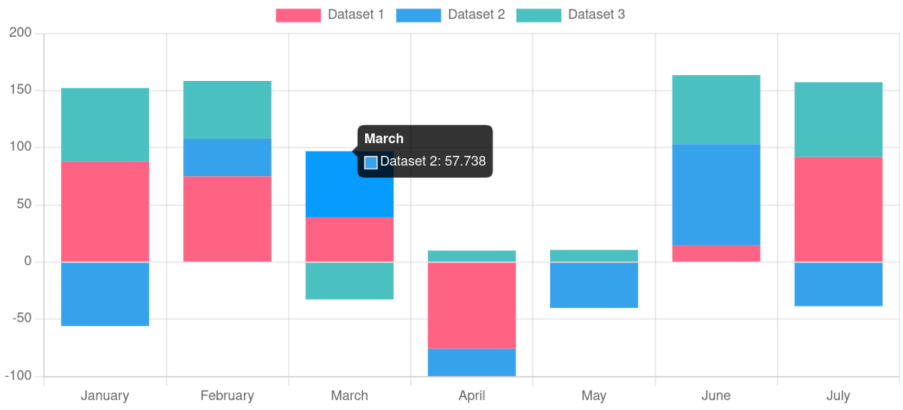
### 2.1.3 Charts.js

Charts.js[3] je populární open-source JavaScriptová knihovna, jejímž cílem je jednoduché vykreslování statických a interaktivních grafů. Je navržena s ohledem na potřeby webových vývojářů a designérů, kteří hledají efektivní nástroj pro vizualizaci dat na svých webových projektech.

#### 2.1.3.1 Prostředí

Knihovna nabízí pouze devět základních grafů, jako jsou spojnicové, sloupcové, plošné či bodové. Výhodou je, že je možné různé druhy grafů kombinovat, čímž se výrazně zvyšuje počet možných zobrazení dat. Vizuální stránku lze v jednoduchých mezích přizpůsobovat, to zahrnuje barvy, popisky, rozložení grafu, legendu a další.

Grafy jsou v základu statické, ale mohou být částečně interaktivní, nebo dokonce i animované, což vylepšuje vizuální stránku grafu a může tak zlepšit uživatelský zážitek. Pokud jsou tyto funkce využity, tak to má bohužel silný dopad na výpočetní náročnost. Tyto funkce jsou na slabších zařízeních, jako jsou starší počítače či mobilní zařízení, spíše na obtíž.




Obrázek 2.4: Charts.js skládaný sloupcový graf [3]

#### 2.1.3.2 Zprovoznění

Charts.js sice vyžaduje určité znalosti fungování webů, ale samotná implementace příliš náročná není. Po připojení knihovny je třeba vytvořit canvas, na který se graf vykreslí, a krátký skript s daty a nastaveními. Proces je jednodušší než u Google Charts, je to však především kvůli tomu, že je k dispozici méně funkcí. Tyto základní funkce lze však rozšířit pomocí pluginů, které přidávají například stínování, gradienty barev či interaktivní legendu.

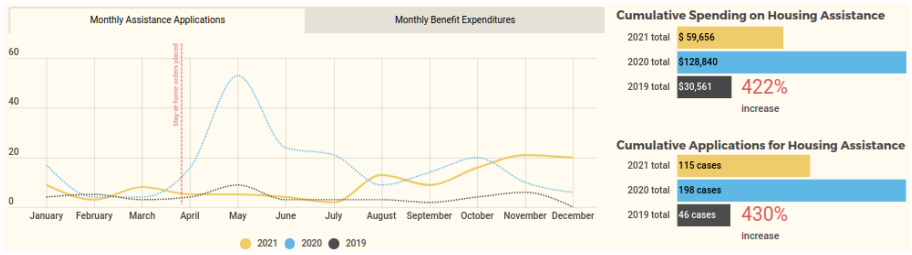
### 2.1.4 Infogram

Infogram[4] je webová služba pro tvorbu interaktivních infografik a datových vizualizací. Je primárně určena pro snadné vytváření interaktivních vizuálních prvků, takže je přívětivá i pro lidi bez programátorských a grafických znalostí. Přestože to není hlavním zaměřením, tak Infogram také poskytuje nástroje a funkce, které umožňují vytvářet atraktivní a funkční grafy pro různé účely.

Jedná se o placenou službu, jež však nabízí také bezplatnou verzi s poměrně širokým spektrem funkcí, které jsou dostatečné pro většinu běžných potřeb. Placené plány zahrnují dodatečné funkce, například více šablon pro tvorbu grafů či statistiky o zobrazeních, interakcích a sdílení.

#### 2.1.4.1 Prostředí

Služba nabízí okolo stovky různých grafů. Toto veliké množství je však vyváženo tím, že jednotlivé grafy umožňují jen velmi základní možnosti úprav. To znamená, že pokud uživatel není spokojen s předdefinovanými možnostmi grafů, nemá možnost je významně upravit. Pro tvorbu projektů Infogram nabízí uživatelům jednoduché a intuitivní grafické rozhraní. Uživatelé mohou pracovat s objekty a ikonami, které mohou snadno přetáhnout a vložit do svých projektů pomocí myši, což zjednodušuje proces tvorby a umožňuje rychle vytvářet profesionálně vypadající vizualizace a infografiky.




Obrázek 2.5: Infogram liniový graf [4]

#### 2.1.4.2 Zprovoznění

Vytvořené vizualizace lze jednoduše vkládat na sociální sítě díky integrovanému rozhraní, případně je možné je vložit na webové stránky jako iFrame či GIF[2]. Data nelze vložit přímo do grafu jako u jiných služeb, místo toho je nutné data importovat do aplikace nebo využít externí databáze. Pro import je možné využít grafické rozhraní nebo jednoduché REST[3] API[4].

### 2.1.5 LiveGap Charts

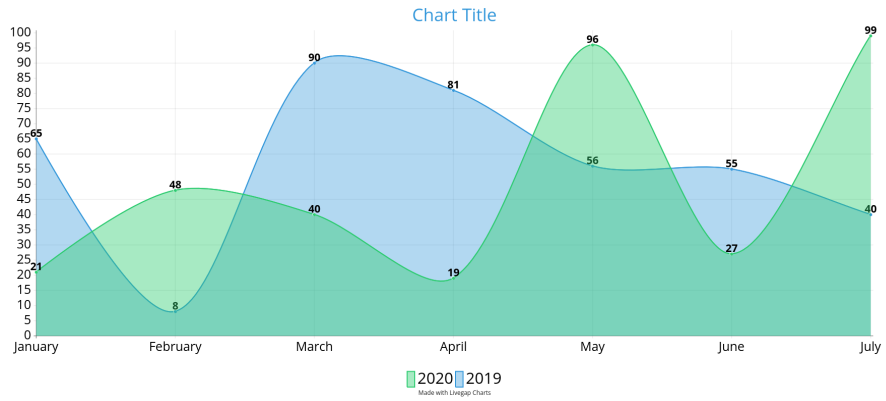
Služba LiveGap Charts[5] umožňuje vytvářet jednoduché grafy bez nutnosti se registrovat. Tato platforma je proto ideální pro ty, kteří hledají jednoduchý a přímý způsob, jak vizualizovat svá data. Pokročilejší funkce jsou placené, ale

[2]Graphics Interchange Format
[3]Representational State Transfer
[4]Application Programming Interface

na rozdíl od podobných služeb je cena pouze pět dolarů měsíčně. Grafy jsou velmi nenáročné na výkon, ale mají jen velmi omezené možnosti interaktivity.

#### 2.1.5.1 Prostředí

Při vytváření grafů v LiveGap Charts je k dispozici uživatelsky přívětivé webové rozhraní, které je snadno ovladatelné i pro začátečníky. V rámci tohoto rozhraní je k dispozici menu pro úpravy vzhledu grafu a jednoduchý tabulkový editor, který umožňuje snadné zadávání a organizaci dat. Uživatelé mohou data vkládat ručně přímo do editoru, nebo je importovat z externích zdrojů pomocí běžných datových formátů, včetně XML[5], CSV[6] a XLS[7].




Obrázek 2.6: LiveGap plošný graf [5]

#### 2.1.5.2 Zprovoznění

Na stránku lze vytvořené grafy vložit několika způsoby. První možností je graf exportovat jako statický obrázek, nebo vytvořit animovaný GIF. Tato metoda má samozřejmě tu nevýhodu, že grafy nelze po exportování upravovat. Pro vložení dynamického grafu do webové stránky je zapotřebí placená verze. Proces je to velmi jednoduchý a vyžaduje vložení pouze jednoho elementu. Nevýhodou však je, že data musí být uložena ve službě LiveGap a není možné je načítat například z externí databáze.

### 2.1.6 Shrnutí

V této kapitole bylo uvedeno pět služeb, které sice všechny slouží k vykreslování grafů, ale jsou zaměřeny na velmi rozdílné cílové skupiny a potřeby uživatelů. Některé z těchto služeb jsou plně zdarma a dostupné pro každého, zatímco jiné nabízejí pokročilé funkce za měsíční poplatky, které mohou dosáhnout i desítek tisíc korun měsíčně. Některé jsou open-source a je možné se podívat i na vnitřní fungování, u ostatních je možné vycházet pouze z toho, k čemu

[5] Extensible Markup Language
[6] Comma Separated Values
[7] Excel spreadsheet

má přístup běžný uživatel. Přestože se jednotlivé služby liší v ceně a zaměření, mají několik společných vlastností.

Všechny služby nabízejí veliké množství grafů, a proto je tato aplikace musí obsahovat také, jinak by měla značnou nevýhodu oproti již zavedeným řešením. Další společnou vlastností je flexibilita v úpravách a přizpůsobení vzhledu grafů. Uživatelé mohou upravovat barevné schéma, styly a další estetické prvky grafů podle svých potřeb a preferencí. Ačkoli se úroveň interaktivity mezi službami může lišit, všechny nabízejí základní interaktivní funkce, jako je zobrazení detailů hodnot po najetí myší.

Služby však také mají i společné nedostatky. Hlavním z nich je vysoká náročnost na výpočetní výkon, což může zpomalit načítání webových stránek a negativně ovlivnit uživatelský zážitek.

## 2.2 Vizualizace dat

Grafy jsou jedním z nejlepších nástrojů pro vizualizaci dat. Zvládnou velmi rychle předat veliké množství dat a umožňují nám vidět vzory, trendy a anomálie, které by jinak mohly zůstat skryty v tabulkách či číselných seznamových formátech. V porovnání s tabulkami jsou grafy často přehlednější a intuitivnější, což umožňuje rychlejší a snadnější interpretaci dat. [10]

Následující sekce stručně uvádí nejčastější druhy grafů, s nimiž se mohou uživatelé praxi nejčastěji setkat. Každý typ grafu má své specifické vlastnosti a výhody, které se hodí pro různé situace a typy dat, a proto je vhodné mít alespoň základní přehled.

### 2.2.1 Bodové

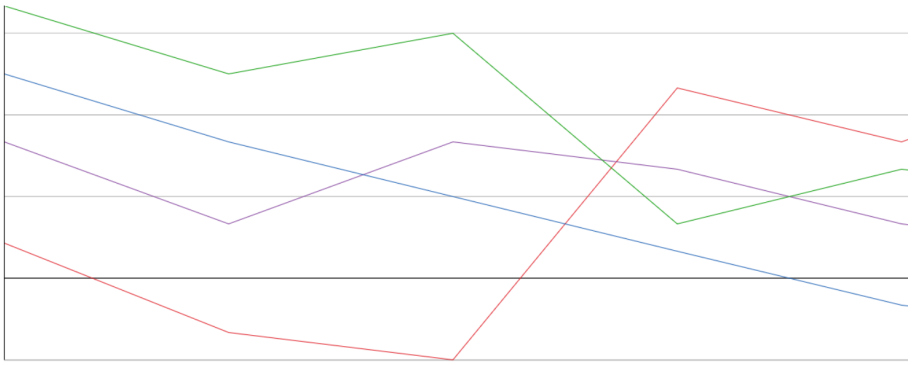
Obrázek 2.7: Bodový graf

Bodové grafy jsou základním nástrojem pro vizualizaci datových bodů na rovině. Každý bod v grafu reprezentuje jednu konkrétní hodnotu nebo pozorování. Tyto datové body mohou být v závislosti na datech rovnoměrně nebo nerovnoměrně rozloženy napříč vodorovnou osou. Tento typ grafu je ideální pro zobrazování vztahů mezi dvěma proměnnými a umožňuje identifikovat vzory, korelace a odlehlé hodnoty. Někdy se jim proto říká korelační diagramy. V bodových grafech jsou jednotlivé body umístěny na kartézské soustavě souřadnic. Mohou být v některých případech i trojrozměrné a každý bod tedy popisují tři

souřadnice, ale v praxi se tato verze příliš nepoužívá kvůli špatné přehlednosti. Bodové grafy jsou často využívány k analýze dat ve vědeckém výzkumu, ekonomii, sociologii a dalších disciplínách, kde je potřeba vizualizovat vztahy mezi proměnnými. [11]

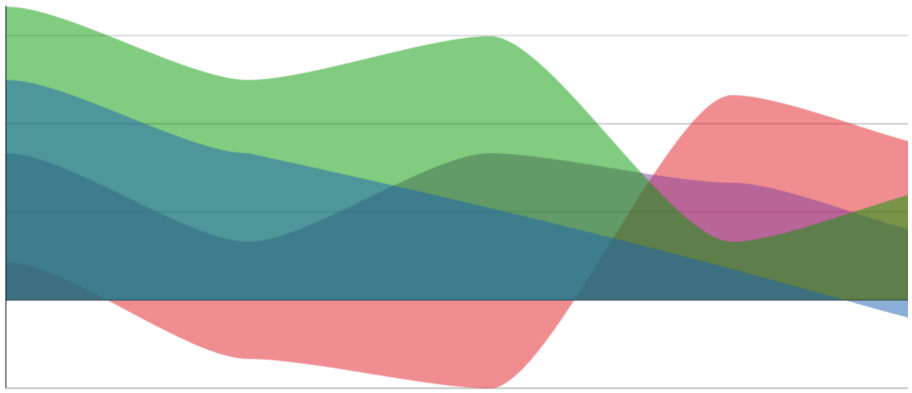
### 2.2.2 Spojnicové

Spojnicové grafy se používají k vizualizaci vztahů mezi body v čase nebo v závislosti na jiné proměnné. Tento typ grafu je vhodný pro zobrazování trendů, vývoje a dynamiky dat. Ve spojnicových grafech jsou body spojeny čarami, které zobrazují vztah mezi nimi. Tyto čáry obvykle představují vývoj dat v čase nebo vztah mezi dvěma proměnnými. Na rozdíl od bodového grafu jsou data téměř vždy rovnoměrně rozprostřena po celé délce vodorovné osy. Spojnicové grafy jsou často využívány k vizualizaci dat jako je vývoj cen, teplot nebo akcií. [11]



Obrázek 2.8: Spojnicový graf

### 2.2.3 Plošné



Obrázek 2.9: Plošný graf

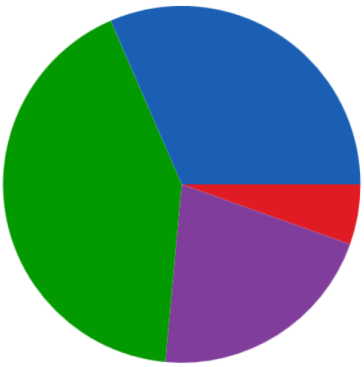
Plošné grafy vizualizují data jako plné oblasti, kde každá oblast reprezentuje jednu datovou kategorii. Tyto grafy jsou ideální pro srovnávání hodnot mezi

různými kategoriemi a pro zjištění jejich relativního podílu. Jednou z hlavních výhod plošných grafů jednoduchost interpretace, protože plné plochy poskytují přehledný obraz o datech a umožňují snadno vizuálně porovnávat velikosti jednotlivých kategorií. Plošné grafy mohou nést i některé nevýhody. Mezi ně patří možná ztráta podrobností, pokud je potřeba zobrazit detailnější informace nebo individuální hodnoty. Mohou být často viděny v oblastech jako marketing, ekonomie nebo třeba ve zdravotnictví, pro srovnání prevalence různých onemocnění.

### 2.2.4 Koláčové

Koláčové grafy jsou kruhové diagramy s různě barevnými výsečemi, které vizualizují data jako procentuální části celku. Každý segment v grafu odpovídá určitému podílu celkového množství. Tento podíl lze vyjádřit buď procenty, nebo přímo hodnotami jako jsou gramy či litry.

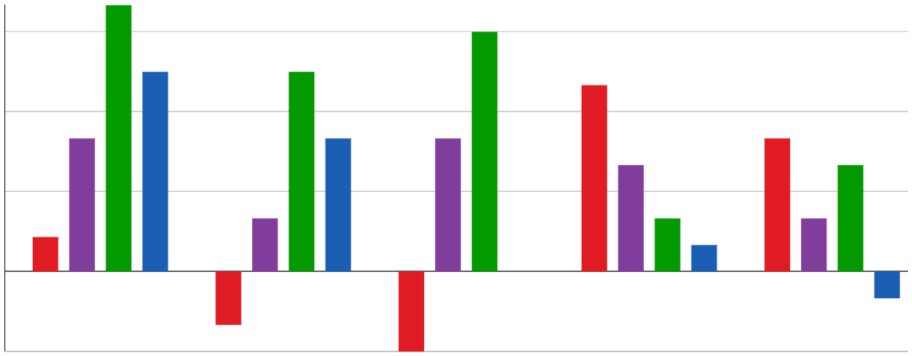
Koláčové grafy jsou často využívány pro prezentaci procentuálního rozložení různých položek, jako jsou například náklady, zisky nebo populace. Jsou také vhodné pro zobrazení struktury portfolia, podílu tržního segmentu a demografických trendů. Mohou však být méně efektivní při zobrazení velkého počtu kategorií a mohou být matoucí, pokud jsou některé segmenty příliš malé na to, aby byly přehledné.



Obrázek 2.10: Koláčový graf

### 2.2.5 Sloupcové

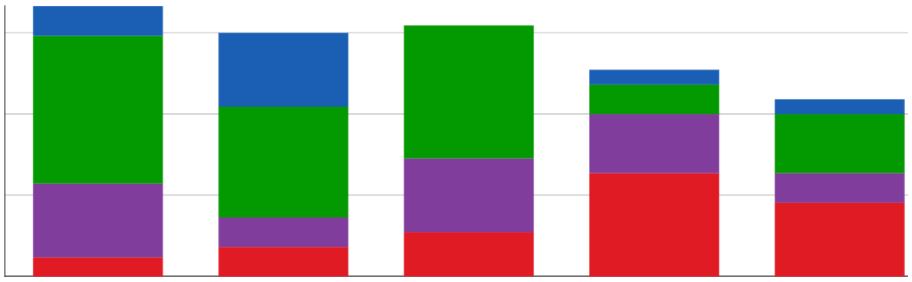
Sloupcové grafy jsou vynikajícím nástrojem pro vizualizaci dat, které mají jasně definované kategorie a hodnoty. Data reprezentují pomocí svislých sloupců, kde výška sloupce odpovídá hodnotě datového bodu. Jsou ideální pro srovnávání hodnot mezi různými kategoriemi a rychlé pochopení relativních hodnot. Tento typ grafu je často využíván v různých oblastech, jako jsou ekonomie, marketing, nebo vědecký výzkum, kde je potřeba srovnávat hodnoty mezi různými skupinami nebo kategoriemi dat. Díky své jednoduché interpretaci a přehlednému zobrazení je sloupcový graf oblíbeným nástrojem nejen pro analyzování dat, ale i pro prezentaci výsledků a komunikaci významných trendů nebo rozdílů.[12]



Obrázek 2.11: Sloupcový graf

### 2.2.6 Skládané

Skládané grafy představují pokročilejší formu sloupcových grafů, kde každý sloupec zobrazuje kumulativní hodnoty všech kategorií namísto jednotlivých hodnot. Skládané grafy jsou užitečné pro porovnání celkové velikosti a struktury různých skupin dat a používají se proto při vizualizaci trendů, jako jsou například příjmy, náklady nebo produkční výstupy.



Obrázek 2.12: Skládaný sloupcový graf

## 2.3 Způsoby vykreslování webových grafů

Existuje mnoho různých způsobů, jak lze interaktivní grafy vytvářet. Každý přístup má své vlastní výhody, avšak i určité nedostatky, ať už se jedná o složitost implementace, nároky na výpočetní výkon nebo omezení v možnostech vizualizace. Tato sekce se zaměřuje na analýzu několika vybraných metod, od historických až po nově zaváděné.

### 2.3.1 Druh grafiky

Počítače dokážou pracovat s rastrovou a vektorovou grafikou. Každá technologie pro vykreslování grafů tak používá alespoň jednu z nich.

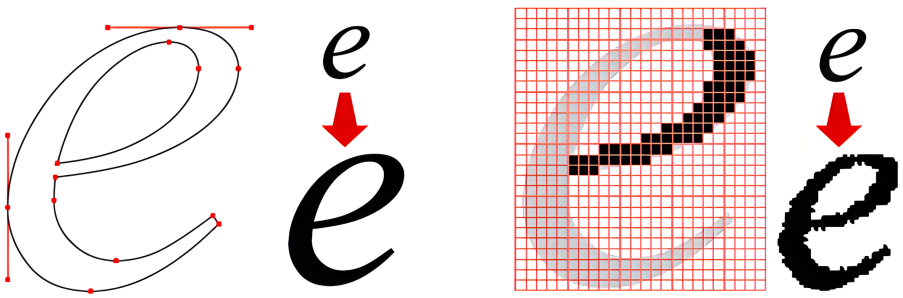
#### 2.3.1.1 Rastrová grafika

Rastrové obrázky jsou ukládány jako mřížka jednotlivých buněk neboli pixelů. Každý pixel má přiřazenou hodnotu určující jeho barvu a intenzitu, což umožňuje vytvářet detailních obrázků. Rastrová grafika je často používána pro fotografie a komplexní obrázky, a to díky své schopnosti zachytit velké množství detailů. Nicméně při zvětšení rastrového obrázku je možné pozorovat ztrátu kvality a zubaté hrany, což může být nevýhodou při manipulaci s obrázky ve velkých rozměrech nebo v případech, kdy potřebujeme přiblížit jednotlivé detaily. Typickými formáty rastrové grafiky jsou JPEG[8], PNG[9] a GIF.[13]

#### 2.3.1.2 Vektorová grafika

Vektorová grafika je typ digitálního obrazu, který je definován pomocí matematických vzorců a vztahů mezi objekty na rozdíl od konkrétních pixelů, jak je tomu u rastrové grafiky. Tento přístup umožňuje vytváření obrázků založených na geometrických tvarech, jako jsou přímky, křivky a základní jednoduché tvary. Díky tomu jsou vektorové obrázky nezávislé na rozlišení a lze je libovolně zvětšovat či zmenšovat bez ztráty kvality.

Vektorová grafika je ideální pro tvorbu log, ikon, diagramů, fontů a dalších grafických prvků, které vyžadují jasnou podobu a čistý vzhled při mnoha případech využití. Nehodí se však pro zachycování komplexních obrázků, jako jsou například fotografie. Obecným formátem vektorové grafiky je SVG a další formáty pro konkrétní případy využití jako například AI pro Adobe Ilustrator.[14]



Obrázek 2.13: Porovnání rastrové a vektorové grafiky [6]

### 2.3.2 Statické grafy

Tato metoda byla využívána již v počátcích HTML[10] a lze na ni často narazit i dnes. Nejednoduší variantou je vložení již vygenerovaného grafu do webové stránky ve formě statického obrázku, přičemž nezáleží na tom, jakým způsobem byl tento obrázek původně vytvořen. Tento přístup nabízí mnoho výhod, jako je například jednoduchá implementace a nízké nároky na výkon klienta

[8] Joint Photographic Experts Group
[9] Portable Network Graphics
[10] Hypertext Markup Language

i serveru. S tím se samozřejmě pojí i značné nevýhody. Je náročné aktualizovat data, jelikož je potřebné vygenerovat a vložit úplně nový graf, webová stránka proto nemůže rychle reagovat na případně změny. Další nevýhodou je minimální možnost interakce na straně webového prohlížeče, protože se jedná o statický, prvek který nedokáže reagovat na akce uživatele. Grafy mohou být špatně čitelné na malých obrazovkách, a naopak na velikých obrazovkách může docházet ke ztrátě kvality.[15] I přes tyto omezení tento klasický přístup stále nachází své uplatnění, zejména v kontextu článků, blogů a jiných statických webových stránek, kde není potřeba často aktualizovat obsah grafu a hlavním cílem je vizuálně zobrazit informace čtenářům.

Další variantou je, že se tento statický graf generuje při načítání stránky. Po převzetí požadavku od od uživatele je tedy na straně serveru vytvořit nový graf z nahraných dat. Může k tomu být využito PHP, Python, Java nebo další jazyky či externí nástroje a aplikace, přičemž může být výsledek rastrový nebo vektorový. Tento přístup umožňuje zaručit, že graf vždy obsahuje aktuální informace. Nicméně, tento přístup může mít i své stinné stránky. Je značně složitější na implementaci než první varianta a generování grafu při každém načtení stránky může významně zvýšit výpočetní náročnost. To může vést k pomalejšímu načítání stránek a zvýšení zátěže serveru, zejména pokud je na stránce více grafů nebo pokud je server zatížen vysokým počtem požadavků.[9] Tyto nedostatky lze částečně vyřešit pomocí cache, kde jsou výsledky na určitou dobu ukládány, aby se stejné grafy nemusely zbytečně generovat vícekrát.

#### 2.3.2.1 HTML image map

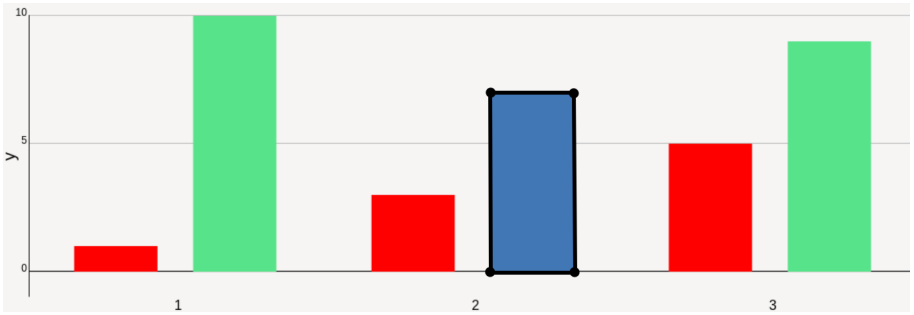
Graf samotný je kompletně statický, ale existují způsoby, jak jim přidat alespoň minimální možnosti interaktivity a zvýšit jejich funkčnost. Jedním takovým je použití prvku MAP, který umožňuje vytvářet interaktivní obrazové mapy na straně klienta a spojuje se s různými prvky, jako jsou IMG, OBJECT nebo INPUT. S jeho pomocí je možné definovat oblasti mapy a přidružené odkazy, což umožňuje uživatelům prozkoumávat obrázky na webových stránkách a provádět s nimi různé akce.

```
<map name="primary">
    <area shape="rect" coords="500,100,60,200"
          href="detail.html" alt="Jablka: 7"/>
</map>
<img usemap="#primary" src="graf.jpg" alt="Sloupcový graf"/>
```

Zdrojový kód 1: ukázka image map

Využívá AREA prvků, které nemají žádný viditelný obsah a pouze vyznačují oblasti na přidruženém prvku, jak je znázorněno na obrázku 2.14. Tyto elementy mohou obsahovat odkazy na další stránky, takže si uživatel může rozkliknout nějakou položku v grafu a následně bude přesměrován na stránku s detailními informacemi.

Dalším využitím je zlepšení přístupnosti webových stránek. S každou oblastí totiž může být spojený text, jak je vidět v ukázce 1, který mohou využívat například čtečky pro nevidomé uživatele a zpřístupnit jim tak alespoň základní informace o grafu.[16]




Obrázek 2.14: HTML image map

### 2.3.3 Pluginy

Jako reakce na omezené schopnosti webových prohlížečů byly vytvořeny nástroje třetích stran, kterým se říká zásuvné moduly neboli pluginy, které měly za cíl tyto nedostatky řešit.

Jedním z hlavních zaměření bylo zajištění dynamických prvků, která dokáží reagovat na akce uživatelů, jako jsou kliknutí, přejetí myší nebo zmáčknutí kláves. Díky těmto funkcím se tyto pluginy staly nepostradatelnými pro tvorbu webových aplikací, online her a interaktivních webových stránek, zejména před zavedením standardu HTML5, který přinesl řadu vylepšení v oblasti interaktivity a multimediálních možností přímo do webových prohlížečů.

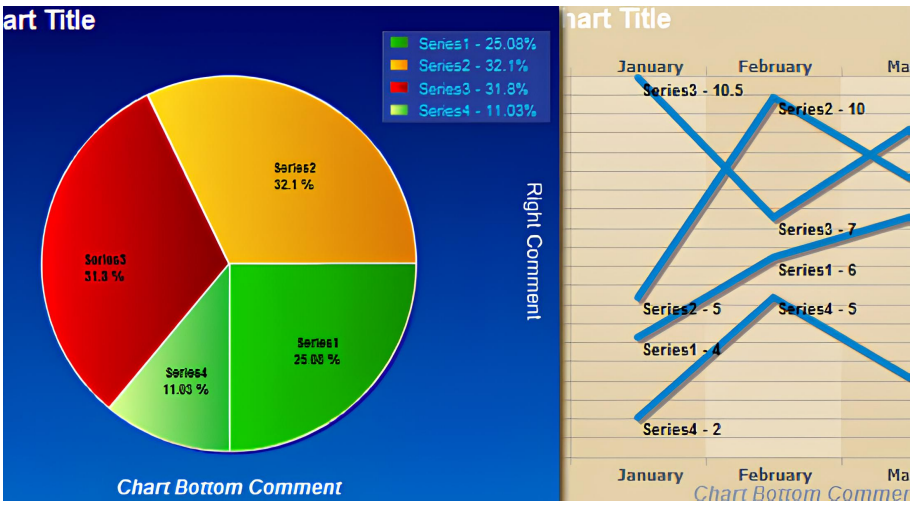
V kontextu interaktivních grafů nabídly tyto zásuvné moduly možnosti, které byly v tehdejší době revoluční. Mimo základních vizualizačních prvků mohly grafy nyní obsahovat animace, zvýrazňování konkrétních hodnot, detailní informace nebo sofistikované filtry pro lepší prohlížení a analýzu dat. Tímto způsobem se interaktivní grafy staly mnohem flexibilnějšími a užitečnějšími nástroji pro prezentaci a interpretaci komplexních dat v online prostředí.[17]

#### 2.3.3.1 Flash Player

Nejznámějším představitelem pluginů je Adobe Flash Player. Byl vytvořen v roce 1996 společností Macromedia, která byla v roce 2006 koupena společností Adobe Systems. Zprvu byl využíván pouze pro přidávání dynamických prvků běžným webovým stránkám jako je zvuk, video či dynamické rozhraní, později se v něm však vytvářely i celé aplikace. V počátcích byl Flash player používán pro přehrávání YouTube videí. Vznikla dokonce nová kategorie her, které by nebylo před vznikem tohoto pluginu vůbec možné vytvořit.

Díky svým rozsáhlým schopnostem si našel využití i při vykreslování webových grafů, jak je vidět na následujícím obrázku 2.15. Jednalo se o univerzální nástroj, takže vzhled grafů nebyl nijak omezen. Existují knihovny, které tuto tvorbu usnadňují. Mohly být i trojrozměrné, což však mělo odpovídající dopad na výpočetní náročnost a rychlost aplikace.

Přestože byl Flash hojně využíván, tak nebyl běžně součástí webových prohlížečů, takže jej bylo potřeba doinstalovat, nebo se smířit s tím, že veliká část funkcí nebude vůbec fungovat. Některá mobilní zařízení Flash nepodporovala a nebylo jej možné ani doinstalovat. Protože se jednalo o další program, který musel být využíván pro prohlížení webu, tak to mělo velmi negativní dopad na




Obrázek 2.15: Graf v Adobe Flash Player [7]

výkon a tedy i rychlost odezvy a načítaní. Začal se proto postupně nahrazovat lepšími řešeními.

Co však nejvíce přispělo k pádu Flash Playeru byla bezpečnost, nebo spíše její nedostatek. Plugin mohl přímo pracovat s prostředky počítače a stal se proto častým cílem útoků. První chyby byly odhaleny v roce 2002. Často nebylo ani nutné aby uživatel prováděl jakékoliv akce, stačilo pouze otevřít danou stránku. I přes známé bezpečnostní chyby byl Flash se záplatami používán až do konce roku 2020, kdy byla oficiálně ukončena podpora.[18]

#### 2.3.3.2 Java applet

Dalším takovým pluginem je Java Applet. Využívá takzvaný bytecode, který je spouštěn v prostředí Virtual Java Machine. Tím se zaručuje přenositelnost kódu, který lze následně spouštět v různých prohlížečích a zařizeních. Aby mohly takovéto aplikace fungovat, tak je často nutné provést instalaci pluginu do prohlížeče. Applety mohou být ovládány i pomocí JavaScriptu, což umožňuje dynamickou interakci s HTML. Při vkládání Java Appletu do webových stránek se původně využíval zastaralý tag <applet>. V současné době je však doporučeno používat modernější tag <object>, který je více standardní a podporován všemi moderními prohlížeči.[19]

Plugin byl vytvořen v roce 1995 společností Oracle a konec podpory byl ohlášen v roce 2017. Byl tedy souběžně s Adobe Flash Playerem, ale nikdy nedosáhl stejného rozšíření a popularity. I přesto zanechává Java Applet svůj odkaz jako jedna z prvních technologií, která umožnila tvorbu dynamického obsahu na webových stránkách a přispěla k růstu a vývoji internetu.[20]

### 2.3.4 Canvas

Canvas je jedním z klíčových prvků HTML5. Jedná se o bitmapovou (rastrovou) oblast na webové stránce, kterou lze dynamicky manipulovat pomocí JavaScriptu. Tento nástroj umožňuje programátorům kreslit různé tvary, text

a obrázky přímo v prohlížeči. Vývojáři mohou využívat barvy, rotace, gradienty a další grafické techniky k vytváření složitých vizuálních efektů. Na webové stránky se tento prvek vkládá pomocí tagu <canvas>, jak je vidět v ukázce 2.

Přistupovat k němu lze pomocí API, které je také součástí HTML5 a umožňuje programátorům plnou kontrolu nad vykreslováním. Díky své flexibilitě a výkonnosti může být canvas v kombinaci s WebGL nebo dalšími renderovacími nástroji využit i pro vykreslování složitých 3D tvarů, interaktivních vizualizací a her, což značně rozšiřuje možnosti webového vývoje. Tato volnost je však vykoupena složitějším ovládáním canvasu. Jedná se o statické plátno, na kterém se obsah vykreslí, veškeré interakce a vykreslování musí být proto řešeny externě pomocí JavaScriptu. Text či jakékoliv jiné prvky nelze po vykreslení upravovat, je potřeba je vykreslit znovu s implementovanými změnami.

Jednou z klíčových výhod canvasu je jeho široká dostupnost, protože je pevnou součástí HTML standardu. To znamená, že je podporován ve všech běžných prohlížečích, což zajišťuje konzistentní zobrazení a funkčnost aplikací napříč různými platformami a zařízeními.[21]

```
<canvas id="canvas" width="200" height="100"></canvas>
```

Zdrojový kód 2: Canvas

### 2.3.5 SVG

SVG (Scalable Vector Graphics) je formát založený na XML určený pro kreslení vektorové grafiky. Vektorová grafika umožňuje detailní a škálovatelné designy, což je ideální pro moderní webové stránky a aplikace.

Stejně jako HTML poskytuje prvky pro definování základních stavebních bloků webové stránky, jako jsou záhlaví, odstavce a tabulky, SVG nabízí prvky pro kreslení základních tvarů. To zahrnuje kruhy, obdélníky, mnohoúhelníky a jednoduché i složité křivky. Každý dokument SVG je strukturován pomocí kořenového prvku <svg>, který definuje oblast pro vykreslování, a základních tvarů, které dohromady tvoří komplexní vektorovou grafiku. Jednoduchý příklad lze vidět v ukázce 3. Kromě toho existuje prvek <g>, který umožňuje seskupení několika základních tvarů do logických celků, což usnadňuje organizaci a manipulaci s grafickými prvky.[22]

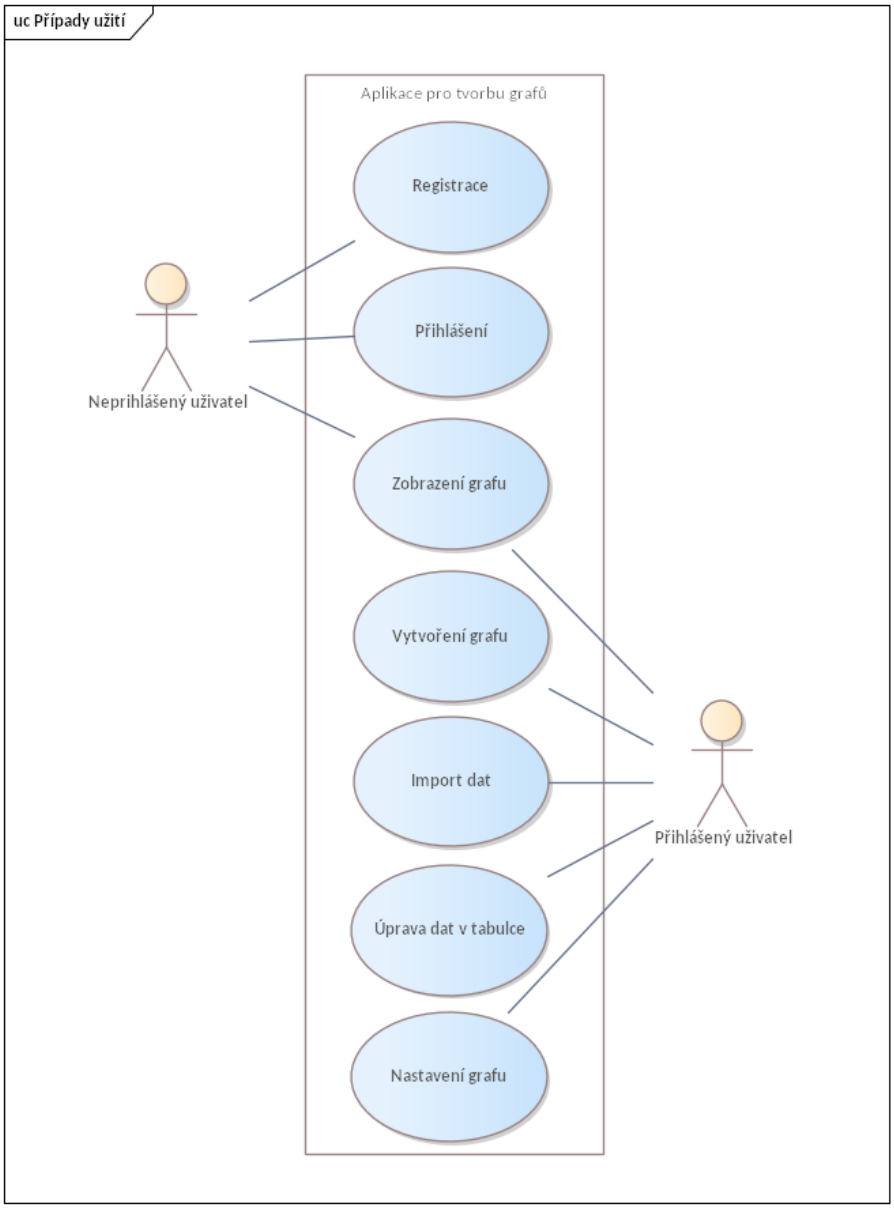
SVG je integrováno do Document Object Modelu (DOM), což umožňuje JavaScriptu přistupovat ke všem objektům a manipulovat s nimi dynamicky. Mohou reagovat na události jako je je pohyb myší, kliknutí a další. To velmi usnadňuje vývoj interaktivních aplikací. Stejně jako v HTML lze i v SVG prvky stylizovat pomocí CSS, což poskytuje vývojářům široké možnosti přizpůsobení vzhledu a stylu grafických prvků. Díky těmto vlastnostem bývá často první volbou pro vykreslování webových grafů, což potvrdila i analýza existujících aplikací. Nevýhodou tohoto přístupu je, že se může pomaleji vykreslovat a má větší nároky na výkon a paměť.[23]

```
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="black" fill="blue"/>
  <text x="30" y="55" font-size="20">SVG</text>
</svg>
```

Zdrojový kód 3: ukázka SVG

## 2.4 Případy užití

1. **Registrace**
   - Uživatelé mohou vytvořit nové účty prostřednictvím registračního formuláře.
   - Při registraci vyplní své osobní údaje, jako je e-mail a heslo.
   - Registrace je realizována pomocí zabezpečeného procesu ověření identity.
2. **Přihlášení**
   - Registrovaní uživatelé se mohou přihlásit pomocí svého uživatelského emailu a hesla.
   - Přihlašovací proces zahrnuje ověření uživatelských údajů pomocí bezpečného protokolu pro přenos dat (HTTPS) a metody hashování hesel.
3. **Vytvoření grafu**
   - Aplikace uživatelům umožňuje vytvářet nové grafy.
4. **Import dat**
   - Aplikace umožňuje uživatelům importovat svá data pro tvorbu grafů ze souboru CSV nebo jiných podporovaných formátů.
   - Import dat je prováděn pomocí robustního mechanismu zpracování souborů.
5. **Upravování dat v tabulce**
   - Uživatelé mají možnost upravovat svá data přímo v tabulkovém editoru v aplikaci.
   - Editor poskytuje pokročilé funkce pro manipulaci s daty, jako je přidávání, mazání a úprava řádků a sloupců.
6. **Nastavení grafu**
   - Uživatelé mohou detailně nastavit různé parametry grafu, jako jsou popisky os, barvy sloupců, rozsah hodnot, volba barevnosti pozadí grafu, zapnutí horizontálního či vertikálního zoomu a další.
   - Nastavení grafu je prováděno prostřednictvím intuitivního uživatelského rozhraní.




Obrázek 2.16: Model případů užití

7. **Zobrazení grafu**

   - Po nastavení parametrů grafu mohou uživatelé zobrazit finální vizualizaci svých dat v podobě grafu přímo v aplikaci.
   - Graf je zobrazován v reálném čase s možností dynamického přizpůsobení.

8. **Vložení grafu na externí webové stránky**
    - Uživatelé mohou vložit svůj vytvořený graf na externí webové stránky pomocí kódu pro vložení (embed code).
    - Tato funkce umožňuje sdílení vytvořených grafů s ostatními uživateli nebo jejich integraci do vlastních webů.
9. **Přístup k aplikaci pomocí API**
    - Aplikace poskytuje uživatelům možnost přístupu k funkcím a datům prostřednictvím API.
    - API umožňuje automatizaci procesů, integraci s dalšími systémy a vývoj vlastních aplikací nebo rozšíření.

## 2.5 Požadavky

Na základě analýzy existujících služeb je sestrojen seznam požadavků, který určuje jednotlivé dílčí cíle práce.

### 2.5.1 Funkční požadavky

- **Vykreslování grafů**
    - Podpora různých typů grafů:
        * Sloupcové
        * Koláčové
        * Prstencové (donutové)
        * Bodové
        * Spojnicové (čárové)
        * Plošné
        * Skládané
        * Radarové (paprskové)
        * Histogramy
        * Kvartilové
        * Stromové
    - **Interaktivita grafů**
        * Možnost zobrazení detailu při najetí myší
        * Zvýraznění dat pomocí interaktivní legendy
        * Zoom a panning pro detailní prohlížení grafů
        * Rozsáhlé možnosti grafických úprav
- **Aplikace na tvorbu grafů**
    - Možnost exportu a sdílení grafů ve formátech jako PNG, JPG
    - Funkce importu a aktualizace dat z CSV, Excelu a dalších formátů
    - Integrovaný tabulkový editor pro manipulaci s daty
    - Možnost vytváření šablon pro opakované použití grafů

- **Správa dat**
  - API pro jednoduchou integraci a správu dat
  - Automatická aktualizace datových sad
  - Možnost připojení k databázím a externím API pro dynamickou aktualizaci dat
- **Uživatelské rozhraní**
  - Intuitivní design a navigace
  - Přizpůsobitelné styly grafů
  - Nastavitelné osy, legenda, barvy, popisky, volba barevnosti pozadí grafu, zapnutí horizontálního či vertikálního zoomu...

### 2.5.2 Nefunkční požadavky

- **Rozšiřovatelnost**
  - Schopnost snadného rozšíření o nové funkce a typy grafů
  - Modulární architektura pro snadnou správu
- **Rychlost**
  - Optimalizace výkonu pro rychlé načítání a vykreslování grafů
  - Efektivní zpracování a vykreslování velkých datových sad
  - Minimalizace zpoždění interakce uživatele s grafy při provádění akcí, jako je přibližování a oddalování
- **Kompatibilita**
  - Podpora všech moderních prohlížečů a operačních systémů
  - Responzivní design pro použití na různých zařízeních (mobilní, tablet, desktop)

### 2.5.3 Priorita požadavků

Tato práce byla zaměřena na to, aby fungovaly všechny základní požadavky a zároveň byla aplikace připravena na jejich další rozšíření. To znamená, že je hlavní implementovat klíčové požadavky, jako je API nebo uživatelské rozhraní.

Není však cílem zahrnout všechny grafy, či formáty souborů pro import, protože budou tyto funkce snadno v budoucnosti implementovatelné díky splnění požadavku rozšiřovatelnosti.

Kapitola 3

# Volba technologií

## 3.1 Vykreslování grafů

Výběr technologie pro vykreslování grafů je jednou z nejvýznamnějších voleb při vývoji této aplikace. Má dopad na výkon, paměťovou náročnost a obtížnost vývoje. Každá technologie se totiž chová jinak než ostatní a ovlivní tak značnou část kódu. Vzhledem k tomu, že se jedná o interaktivní grafy, tak je možné z výběru rovnou vyřadit technologii statických grafů. Tím zbývají Canvas, Pluginy a SVG. Aby se zajistila co největší kompatibilita a jednoduchost používání, tak není vhodné využívat pluginy třetích stran a na výběr tedy zbývají pouze Canvas a SVG, které jsou součástí HTML5 standardu. Obě technologie mají svá výrazná pro i proti, a je tedy nutné zvážit, která z nich se nejlépe hodí pro tuto bakalářskou práci.

Nejprve se podíváme na SVG. Grafy se obvykle skládají ze základních geometrických tvarů, pro jejichž vykreslování je tato technologie stvořena. Zároveň se SVG objekty snáze ovládají a může se s nimi pracovat i po jejich vykreslení a HTML obsahuje prostředky pro interakci s nimi. Z těchto důvodů využívá valná většina existujících řešení právě tuto grafiku. Ačkoliv by se na první pohled mohlo zdát, že vektorová grafika je v tomto porovnání jasný vítěz, není tomu tak. Problémem je ohromné množství potřebných objektů při větším množství zobrazovaných dat. To vede k větším nárokům na výkon zařízení a k pomalému vykreslování grafů.

Jak je na tom tedy canvas? Namísto vykreslování jednotlivých objektů, jako je tomu u SVG, se používá kreslící plocha, na kterou jsou vykreslovány obrazy, tvary a text. To umožňuje efektivnější vykreslování grafů s velkým množstvím dat, protože se vykreslují pouze pixely na plátně, nikoli jednotlivé objekty. Tento přístup může vést k lepšímu výkonu a rychlejšímu vykreslování, zejména při práci s velkým objemem dat.

## 3.2 Frontend

Frontend webové aplikace je zodpovědný za uživatelské rozhraní a interakci s uživateli. Jeho účelem je prezentace obsahu a komunikace s backendem. Díky frontendu mohou uživatelé snadno navigovat, provádět akce a využívat funkce aplikace.

### 3.2.1 HTML

HTML (HyperText Markup Language) je základním stavebním kamenem webových aplikací, který definuje význam a strukturu prezentovaného obsahu. Hypertext je způsob strukturování textu, který využívá takzvaných hyperlinků, neboli odkazů na další dokument v rámci jedné nebo více webových stránek. Vzniká tak síť provázaných dokumentů, které jsou základem moderního webu.

HTML využívá specifické značky, známé jako tagy, k označení a organizaci různých typů obsahu, jako jsou textové odstavce, obrázky, odkazy a další pro zobrazení ve webovém prohlížeči. Běžnými příklady jsou <p> pro označení odstavců, <head> pro metadata a mnoho dalších. Běžně se používá v kombinaci s CSS[11], které určuje vzhled stránky, a s JavaScriptem, který zpracovává funkcionalitu a dynamické chování. [24]

### 3.2.2 CSS

CSS (Cascading Style Sheets) je jazyk pro určování vzhledu HTML a XML dokumentů. Tento jazyk byl vytvořen jako reakce na potřebu oddělení obsahu a designu, což výrazně usnadňuje údržbu, aktualizaci a rozšíření webových stránek. Je založen na standardu společnosti W3C[12].

Při použití samotného HTML budou stránky používat stejný font, velikost písma, barvu elementů, odsazení a další vizuální vlastnosti. Pomocí CSS lze tyto vlastnosti nastavit a určit tím tak vzhled webové stránky. CSS je hierarchický a kaskádový jazyk, což znamená, že styly se dědí od rodičovských prvků a mohou být přepisovány pomocí specifičtějších selektorů. Díky tomu umožňuje snadnou aplikaci globálních stylů na celou webovou stránku, zatímco zároveň poskytuje možnost vytvoření specifických stylů pro individuální prvky. Styly lze vkládat přímo do HTML dokumentů nebo do externích souborů. Toto oddělení vzhledu od obsahu také zlepšuje výkon webových stránek tím, že umožňuje prohlížečům načíst styly paralelně s obsahem, což zkracuje dobu načítání a zlepšuje uživatelský zážitek. [25]

### 3.2.3 JavaScript

JavaScript je multiplatformní objektově orientovaný skriptovací jazyk, který se používá k vytváření interaktivních webových stránek. Může být interpretovaný, nebo JIT[13] kompilovaný pro vyšší výkon. Nejznámější je jako skriptovací jazyk pro webové stránky, ale používá se i v mnoha dalších prostředích mimo prohlížeč, jako jsou například Node.js, Apache CouchDB a Adobe Acrobat.

Přestože byl původně vytvořen pro frontend, ale lze pomocí něj vytvářet systémy i na straně serveru. JavaScript se používá pro vytváření různých funkcí a efektů na webových stránkách, jako jsou animace, validace formulářů, dynamická aktualizace obsahu bez nutnosti obnovení stránky, AJAX[14] pro komunikaci s webovými servery a mnoho dalšího. Díky těmto vlastnostem JavaScript umožňuje vývojářům vytvářet responzivní a interaktivní webové aplikace. [26]

[11] Cascading Style Sheet
[12] World Wide Web Consortium
[13] Just-in-time
[14] Asynchronous JavaScript and XML

### 3.2.4 Shrnutí

Kombinace HTML, CSS a JavaScriptu je zlatým standardem při vývoji webových aplikací. Ačkoliv existují i další technologie a frameworky, které nabízejí podobné funkce a možnosti, ale ty stále nejsou podporované na všech zařízeních a internetových prohlížečích. Proto jsou tyto tři technologie zvoleny pro vývoj této webové aplikace.

## 3.3 Backend

Backend webové aplikace je část systému, která zpracovává data a logiku na straně serveru. Zajišťuje správu databáze, zpracování požadavků od uživatelů a další úkoly. Volba technologií má veliký dopad na výkon, kompatibilitu a škálovatelnost aplikace.

### 3.3.1 PHP

PHP (Hypertext preprocessor) je populární skriptovací jazyk, který je určený především k vývoji dynamických webových stránek a aplikací. Vznikl v devadesátých letech pro účely osobní domácí stránky (původní jméno Personal Home Page), od té doby prošel mnohou změn. Dnes se jedná o flexibilní jazyk s širokým spektrem funkcí a možností použití. Lze jej integrovat přímo do HTML, kde může sloužit k drobnému vytváření obsahu, nebo může být využit jako základ celé webové aplikace a plnit množství úloh jako je například komunikace s databází. Podporuje funkcionální programování, OOP[15] a další přístupy. Veškerý kód se vyhodnocuje na straně serveru bez předchozího kompilování. Díky rozsáhlé komunitě a podpoře pro veliké množství platforem zůstává PHP jedním z hlavních nástrojů pro tvorbu webových aplikací. Proto také bývá PHP často používáno v rámci webhostingů. [27]

Základní funkcionalitu PHP lze rozšířit pomocí frameworků, které usnadní a urychlí vývoj webových aplikací. Jedním takovým je například Symfony. Jedná se o open-source framework, který poskytuje širokou škálu nástrojů a komponent pro zjednodušení běžných úkolů, jako je správa routování, práce s formuláři, manipulace s databází a řízení uživatelských oprávnění. [28]

Další funkce lze přidat pomocí rozsáhlého ekosystému knihoven. Je vhodný pro projekty různých velikostí a komplexity, od malých webových stránek až po rozsáhlé aplikace.

### 3.3.2 Java

Název Java může popisovat buď programovací jazyk nebo vývojové a runtime prostředí. Byl vytvořen v roce 1995 firmou Sun Microsystems a od té doby získal širokou popularitu díky své flexibilitě a přenositelnosti. Java je objektově orientovaný jazyk, který se odlišuje od tradičních kompilovaných jazyků tím, že nevytváří strojový kód přímo, ale generuje tzv. bytecode, který je následně spouštěn na JVM[16]. Díky tomu je program velmi dobře přenositelný a lze jej spouštět na mnoha rozdílných platformách a systémech. Java poskytuje mnoho

[15] Objektově Orientované Programování
[16] Java Virtual Machine

technologií pro webové aplikace. Příkladem je JSP[17] pro dynamické vytváření obsahu. [29][30]

Kromě toho existují různé populární frameworky a nástroje, jako je například Spring Framework, který poskytuje komplexní sadu nástrojů pro tvorbu webových aplikací, komunikaci s databází a mnoho dalších funkcí. Spring open-source projekt, který má rozsáhlou a aktivní komunitu, což zaručuje jeho neustálý vývoj a podporu. Podporuje mnoho způsobů využití od drobných cloudových aplikací až po systémy velikých firem. [31]

### 3.3.3 Node.js

Node.js je open-source, cross-platformovní, JavaScriptové runtime prostředí, které umožňuje vývojářům psát backend webových aplikací v JavaScriptu. Umožňuje spouštění kódu mimo webový prohlížeč a umožňuje asynchronní a událostmi řízené programování. Když Node.js provádí operaci, jako je načítání dat ze sítě, či přístup k databázi nebo souborovému systému, tak proces zahájí až když obdrží odpověď, místo toho aby aktivně čekal a plýtval cykly CPU. To zajišťuje efektivní zpracování více požadavků současně. Hlavní výhodou však je, že se vývojáři nemusí učit více jazyků a mohou vytvářet jak frontend tak i backend pouze pomocí JavaScriptu. Node.js je také optimalizován pro práci s nejrůznějšími databázemi, souborovými systémy a sítěmi. Nabízí bohatou sadu modulů a nástrojů, které usnadňují integraci a vývoj aplikací v různých oblastech, včetně IoT, real-time komunikace a cloudových služeb. [32]

Vývojáři mohou využít rozsáhlou knihovnu Node.js, zvanou npm (Node Package Manager), která obsahuje tisíce balíčků a modulů připravených k použití. To značně urychluje vývoj a umožňuje vývojářům vytvářet komplexní aplikace s minimálním úsilím. [33]

### 3.3.4 Shrnutí

Pro vývoj této webové aplikace bude využito PHP v kombinaci s frameworkem Symfony. Jedná o nejvíce používaný jazyk pro backend webových aplikací a má díky tomu nejvíce pomocných materiálů a návodů. Java a Node.js také mají své silné stránky, které však nenajdou využití vzhledem k menšímu rozsahu webové aplikace. PHP má navíc nespornou výhodu, že je kompatibilní s většinou webhostingů, a velmi tak usnadní finální nasazení aplikace.

## 3.4 Databázové technologie

Databáze slouží k ukládání, načítání a správě dat. Tyto technologie umožňují webovým aplikacím uchovávat uživatelské údaje, obsah stránek, nastavení, historii transakcí a mnoho dalšího. Jsou základem mnoha webových aplikací, jejichž účelem je především prezentace dat uživateli.

### 3.4.1 MySQL

MySQLje jedním z nejpopulárnějších a nejvíce používaných relačních databázových systémů na světě. Vývoj, distribuci a podporu zajišťuje Oracle Corpo-

[17] JavaServer Pages

ration. Hlavním principem MySQL je ukládání dat do strukturovaných tabulek, které lze mezi sebou propojit pomocí primárních a cizích klíčů, což umožňuje efektivní a organizovanou správu dat. Pro vytváření dotazů a ovládání databáze využívá jazyk SQL[18].

Jedním z klíčových aspektů MySQL je jeho zaměření na bezpečnost a integritu dat. Databázový systém MySQL implementuje pokročilé zabezpečení a kontrolní mechanismy, které zajišťují, že data jsou chráněna a zachována v neporušeném stavu během provádění transakcí a operací. Dále MySQL nabízí vysokou škálovatelnost a flexibilitu, což umožňuje efektivní správu dat a zpracování vysokého provozu. Tento systém je vhodný pro širokou škálu aplikací, od malých a středních webových stránek, e-shopů až po komplexní podnikové systémy a databázové řešení s vysokým objemem dat. [34]

### 3.4.2 PostgreSQL

PostgreSQL je open-source relační databázový systém, který se vyznačuje svou robustností a flexibilitou. Je navržen tak, aby podporoval širokou škálu datových typů, od běžných relačních dat až po moderní formáty jako JSON[19], XML a geometrická data. Tento databázový systém nabízí pokročilé funkce včetně pohledů, triggerů, procedur a indexů, které umožňují efektivní správu a manipulaci s daty. Důležitou vlastností PostgreSQL je jeho schopnost provádět operace nad daty v rámci transakcí s atomickým, konzistentním, izolovaným a trvalým chováním (ACID[20]), což zajišťuje spolehlivost a integritu dat. Navíc podpora pro externí rozšíření rozšiřuje možnosti systému a umožňuje integrovat další funkcionality a nástroje.

Díky těmto vlastnostem se PostgreSQL stává oblíbenou volbou pro vývojáře a organizace různých velikostí. Jeho všestrannost umožňuje využití v menších webových aplikacích, ale také v komplexních analytických systémech a rozsáhlých enterprise aplikacích, kde je potřeba spolehlivého a výkonného řešení pro správu a ukládání dat. [35]

### 3.4.3 MongoDB

MongoDB je moderní dokumentově orientovaný NoSQL databázový systém, který je navržen pro snadné ukládání a manipulaci s velkými objemy nestrukturovaných dat. Databáze MongoDB využívá schématu ukládání dat ve formě dokumentů v JSON formátu, což je nativní pro mnoho programovacích jazyků, včetně JavaScriptu, což usnadňuje integraci a manipulaci s daty. Díky tomuto přístupu zvládá ukládat různé typy nestrukturovaných dat, což zahrnuje textové dokumenty, obrázky, videa a další multimediální obsah. Dále MongoDB nabízí robustní možnosti horizontálního škálování a replikace, což umožňuje distribuované ukládání dat a efektivní správu vysokého provozu.

Díky těmto funkcím poskytuje MongoDB vysokou dostupnost a odolnost vůči selháním, což zajišťuje nepřetržitý provoz aplikací i v případě výpadků komponent. V kombinaci s těmito vlastnostmi MongoDB poskytuje flexibilní a škálovatelné řešení pro různé aplikace, od webových služeb, mobilních aplikací

[18] Structured Query Language
[19] JavaScript Object Notation
[20] Atomicity, Consistency, Isolation, Durability

až po velké podnikové systémy, kde je potřeba rychlého, spolehlivého a odolného databázového řešení. [36]

### 3.4.4 Shrnutí

Hlavním účelem je zobrazování grafů a databáze bude tedy především data pouze uchovávat a vypisovat. Nebudou zde probíhat transakce mezi jednotlivými grafy, a není proto nutné primárně řešit zachování integrity dat, jako by tomu bylo například u bankovního systému. Data, která budou v databázi uložena, budou pouze částečně strukturovaná a je nutné počítat s možným přidáváním dalších nabízených grafů. Dokumentová databáze se tedy jeví jako ideální řešení pro tento typ aplikace, protože umožňují rychlé a jednoduché ukládání a získávání dat v podobě dokumentů, které odpovídají struktuře používaných grafů. Nejlepší volbou pro účely této aplikace je tedy MongoDB.

Kapitola 4

# Návrh

Tato kapitola se věnuje návrhu jednotlivých částí aplikace na základě dříve provedené analýzy. Hlavním cílem projektu je poskytnout nástroj pro vykreslování grafů, který lze jednoduše integrovat do webových stránek. Pro tyto účely je vhodné využít modulární architekturu aplikace. Modularita umožňuje oddělení části aplikace zodpovědné za vykreslování grafů od ostatních funkcionalit, jako je například backend nebo databáze. Tímto způsobem můžeme snadno rozšiřovat, upravovat a spravovat každou část aplikace nezávisle na ostatních. Díky modularitě je také možné implementovat nové funkce nebo změnit již existující bez výrazných dopadů na ostatní části systému. Takový přístup umožňuje vysoce flexibilní a udržitelný vývoj aplikace, která může snadno reagovat na budoucí požadavky a změny v prostředí.

## 4.1 Vykreslování grafů

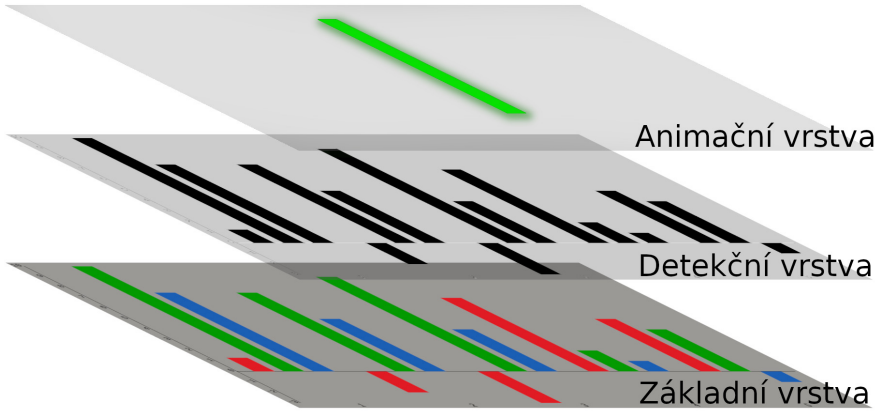
Účelem této části je načítání dat, jejich vykreslování ve formě grafů a zajištění interaktivity pro uživatele. Pro rychlejší vykreslování se doporučuje jej rozdělit na více částí a vrstev. Protože canvas nenabízí možnost vrstev, jako je tomu například u grafických programů jako GIMP[21], tak je nutné využít více canvas elementů nad sebou. V případě této aplikace se jedná o základní vrstvu, na které se nachází všechny potřebné informace, vrstvu zajišťující interaktivitu a vrstvu s animacemi a efekty. [37]

### 4.1.1 Základní vrstva

Základní vrstva je hlavní vrstvou grafu. Zahrnuje kompletní vykreslený graf ve zvolených barvách včetně titulku, os, popisků, pomocných čar a dalších vizuálních prvků potřebných k prezentaci dat.

Tato vrstva tvoří statický základ grafu, jelikož její vykreslování je nejnáročnější částí procesu. Na základní vrstvě jsou vidět všechny interaktivní prvky, avšak interaktivita samotná probíhá jinde. Vrstva se však musí dynamicky měnit při použití funkcionality, jako je zoom nebo posunutí.

[21] GNU Image Manipulation Program




Obrázek 4.1: Znázornění vrstev grafu

### 4.1.2 Detekční vrstva

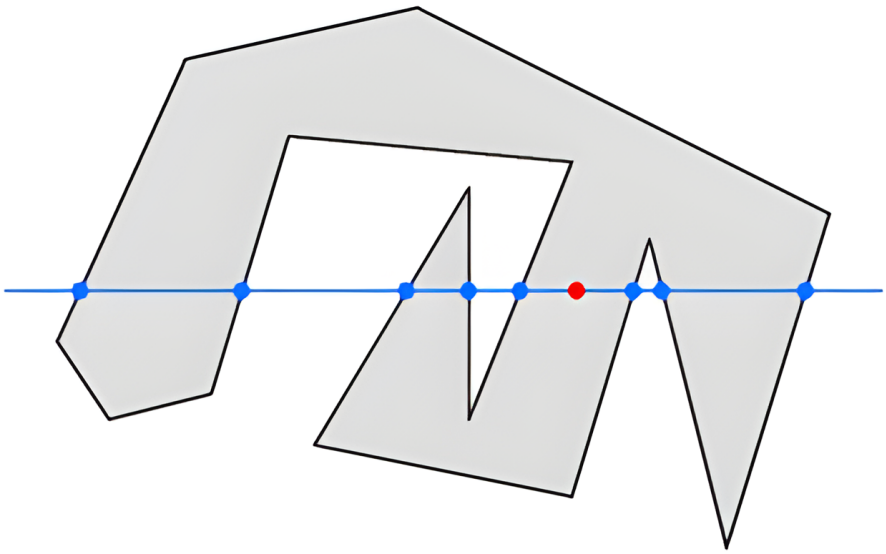
Účelem této vrstvy je zajištění interaktivity. canvas neobsahuje žádné objekty, které by bylo možné manipulovat a reagovat na vyvolané události, jako je například kliknutí myší. Vstup od uživatele je tedy nutné zpracovat jiným způsobem. Existuje několik možností jak toho docílit.

První možností je, že celý canvas reaguje na akce uživatele. Vykreslené objekty mohou být uloženy i se všemi svými parametry, jako je pozice, tvar, velikost a podobně. Při kliknutí nebo jiné události lze načíst momentální pozici myši a převést jí na souřadnice na canvasu. Poté se pozice myši porovná se všemi uloženými objekty. Pro obdélník to znamená, že pokud se souřadnice myši nachází mezi levým horním a pravým dolním rohem, tak se nachází uvnitř obdélníku. Pro kruh by se porovnávala vzdálenost kurzoru myši od středu kruhu, aby se nacházela uvnitř, pak musí tato vzdálenost být menší než poloměr daného kruhu. Takto lze zkontrolovat všechny vykreslené tvary, ale u mnohoúhelníků a podobných je tento proces značně složitější, viz 4.2.

Tento přístup má své výhody, jako je jednoduchost implementace, možnost překrývajících se tvarů a rychlost načtení grafu. Pro malé množství dat je ideální, ale z přibývajícími hodnotami začne být patrná hlavní nevýhoda. Vzhledem k tomu, že je nutné porovnat polohu kurzoru se všemi vykreslenými tvary, tak výpočetní náročnost stoupá lineárně. na slabších zařízeních je citelné zpoždění mezi uživatelskou akcí a reakcí grafu. Lze samozřejmě provádět optimalizace, jako je rozdělení plochy na zóny, které umožní porovnávání omezit jen na část objektů. Tím se však značně zesložiťuje implementace a jádro problému to stejně nevyřeší.

Druhou možností je využití image mapy, která byla zmíněna předcházející kapitole. Umožňuje na obrázku vymezit zóny, pomocí nichž lze následně díky JavaScriptu získávat vstup od uživatele. Tato metoda však sdílí nevýhody s SVG, jako je veliký počet objektů v DOM a pomalé načítání, takže se nehodí pro účely této aplikace, která má za cíl minimalizovat nároky na výkon.

Bylo by ideální, kdyby existoval způsob, jak přesně detekovat, s čím chce uživatel interagovat, který by měl rychlou odezvu a nevyžadoval by příliš prostředků zařízení. Zde přichází na řadu detekční vrstva. canvas API umí předat



Obrázek 4.2: Detekce bodu uvnitř polygonu [8]

informace o konkrétních pixelech, takže je možné zjistit, co přesně se pod kurzorem myši nachází.

Pro každý prvek na základní vrstvě, který má být interaktivní se na detekční vrstvu vykreslí jeho kopie. Každý má svou unikátní barvou, jejíž hodnota odpovídá indexu daného prvku. Každý barevný kanál poskytuje 256 hodnot, maximální počet unikátních objektů je tedy 16 777 216. Může se to zdát jako omezení, ale je to více než dostatečné, protože běžné monitory ani nemají tolik jednotlivých pixelů.

### 4.1.3 Animační vrstva

Tato vrstva slouží k dynamickému vykreslování specifických prvků grafu v reakci na akce uživatele. Když uživatel provede interakci, jako je kliknutí na prvek grafu či přejetí myší, tak animační vrstva reaguje tím, že vykreslí vybrané prvky s využitím animačních efektů. Tyto efekty mohou zahrnovat zvýraznění, plynulé zvětšení nebo zmenšení prvků, barevné změny či jiné vizuální úpravy.

Důvodem samostatné vrstvy je, že vykreslování jednotlivých prvků je výpočetně náročné, a je proto vhodné aby se jich vykreslovalo co nejméně. Toho je docíleno oddělením animací a efektů od zbytku grafu. Při každé interakci tedy stačí vykreslit pouze několik málo objektů, jak lze vidět na obrázku 4.1, a velmi se tím zrychlí odezva grafu, což vede k příjemnějšímu uživatelskému zážitku.

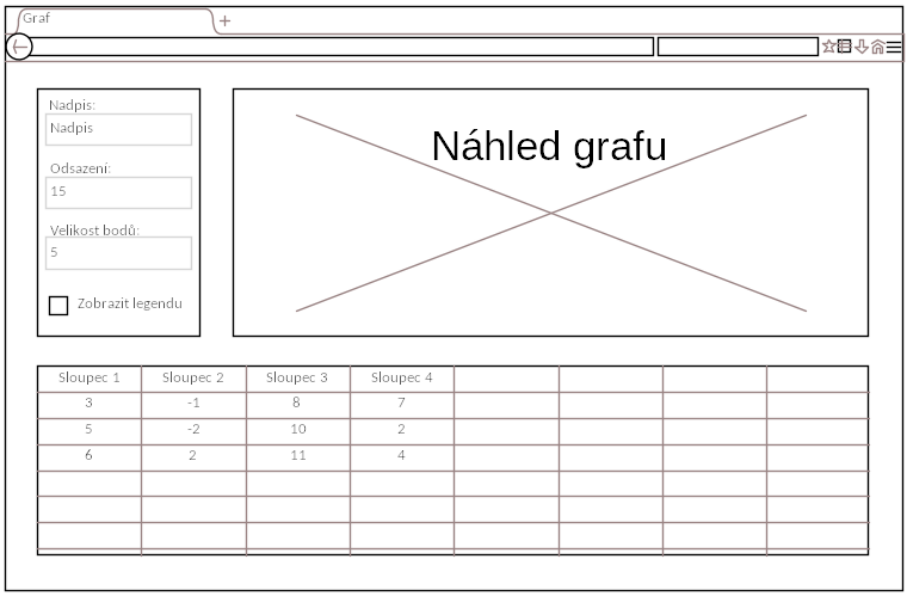
## 4.2 Aplikace na vytváření grafů

### 4.2.1 Stránka pro úpravu grafu

Hlavní částí GUI[22] aplikace je stránka pro úpravu grafů a dat. Na obrázku 4.3 je jednoduchý wireframe diagram, který ilustruje, jak by tato stránka měla vypadat. Dolní část okna je vyhrazena pro tabulkový editor, kde uživatelé mohou vkládat a upravovat data, která budou použita pro vytvoření grafu. Tento editor umožňuje uživatelům zadávat hodnoty a upravovat je v reálném čase, přidávat či odebírat sloupce a řádky nebo nastavovat jméno datové kategorie, což značně usnadní proces tvorby grafu. Bez tabulkového editoru by bylo nutné data připravovat externě a poté je v již finální podobě importovat.

V levém horním rohu se nad tabulkou nachází panel s nastaveními. Je určen pro správu všech vizuálních vlastností grafu, od barvy pozadí a titulku až po kontrolu zoomování a dalších interaktivních funkcí.

Třetím důležitým prvkem je okno s vykresleným grafem, které slouží k vizuální kontrole vzhledu grafu. Uživatelé zde mohou vidět náhled grafu na základě dat z tabulky a nastavení z panelu, což jim umožňuje okamžitě vidět výsledek své práce a případně provádět další úpravy.




Obrázek 4.3: Rozložení stránky pro vytváření grafů

### 4.2.2 Uživatelské účty

Pro zajištění osobního přístupu a možnosti uchovávání dat je aplikace vybavena systémem uživatelských účtů. Uživatelé mají možnost vytvořit si svůj vlastní účet, který jim umožňuje ukládat svá data a nastavení. Díky uživatelským účtům mají uživatelé možnost přistupovat ke svým datům z libovolného zařízení s připojením k internetu a pracovat s nimi kdykoliv a odkudkoliv.

[22] Graphical User Interface

Uživatelské účty zajišťují také bezpečnost dat a soukromí uživatelů prostřednictvím autorizovaného přístupu a ochrany heslem.

## 4.3 API

API (Application Programming Interface) je nástrojem pro interakci mezi softwarovými aplikacemi a službami, poskytujícím sadu funkcí, procedur a protokolů, které umožňují programátorům přistupovat k funkcím a datům jiných systémů. Slouží k tomu, aby umožnil komunikaci a výměnu dat mezi různými systémy, a poskytuje tak programátorům možnost využít již existujících řešení a integrovat je do svých vlastních aplikací. API tak zjednodušuje vývoj nových aplikací a umožňuje rychlou a efektivní integraci nových funkcí a datových zdrojů.[38]

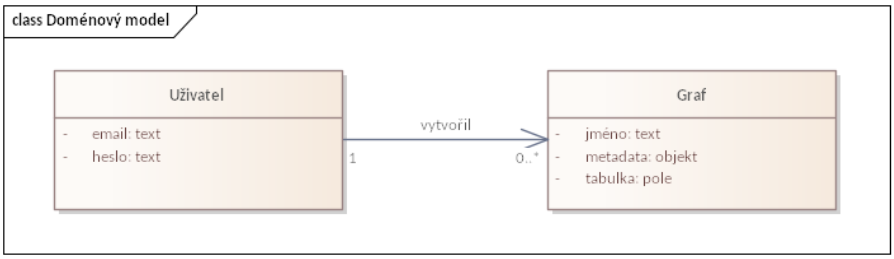
V rámci tohoto řešení je API využito pro nahrávání dat na server a jejich načítání, což umožňuje dynamické aktualizace a vytváření grafů s nejnovějšími informacemi. Například je možné vytvořit webovou stránku, která zobrazuje teplotu za posledních 24 hodin a pomocí API automaticky načítá aktuální data.

Obsahuje endpointy pro registraci a přihlášení uživatelů a pro vytváření grafů, aktualizaci dat a načítání potřebných informací. Jedná se proto o REST API.

## 4.4 Databáze

Tato aplikace se soustředí na vykreslování a tvorbu grafů, stačí tedy ukládat pouze data grafů a uživatelské profily. Díky tomu může být celý databázový model velmi jednoduchý.

### 4.4.1 Doménový model




Obrázek 4.4: Doménový model

#### 4.4.1.1 Uživatel

Entita Uživatel představuje základního aktéra v systému, který je schopen vytvářet a manipulovat s grafy. Každý uživatel má unikátní identifikátor, e-mailovou adresu a heslo, které slouží k autentizaci a zabezpečení jeho účtu.

- **id**: Jednoznačný identifikátor uživatele v systému, používaný pro identifikaci a provázání s dalšími daty v databázi.
- **email**: E-mailová adresa uživatele, která slouží jako unikátní identifikátor pro přihlášení a komunikaci s uživatelem.
- **heslo**: Heslo uživatele, které je hashováno a ukládáno v databázi, aby byla zajištěna bezpečnost účtu. Při přihlašování je heslo porovnáváno s uloženým hashem pro ověření identity.

#### 4.4.1.2 Graf

Entita Graf reprezentuje vizuální prezentaci dat v systému, která může být vytvořena a upravována uživateli. Graf obsahuje informace o svém identifikátoru, názvu, metadatech a tabulce, která obsahuje data potřebná pro jeho vizualizaci.

- **id**: Unikátní identifikátor grafu, který slouží k jeho jednoznačné identifikaci v systému a je používán například v URL pro odkaz na konkrétní graf.
- **jméno**: Název grafu, který slouží k identifikaci a popisu jeho obsahu. Jméno by mělo být výstižné a informativní, aby uživatel okamžitě rozuměl obsahu grafu.
- **metadata**: Informace o grafu jako nadpisy, odsazení, typ grafu, barva pozadí, font legendy a další vizuální vlastnosti.
  - *nadpisy*: Slouží k označení jednotlivých částí grafu, například osy x a y nebo nadpis grafu.
  - *odsazení*: Určuje prostor mezi okrajem grafu a jeho obsahem.
  - *typ grafu*: Definuje, jakým způsobem jsou data vizualizována, například spojnicový, sloupcový, koláčový nebo bodový graf.
  - *barva pozadí*: Specifikuje barvu pozadí grafu.
  - *font legendy*: Určuje vzhled textu v legendě grafu, například velikost a styl písma.
  - *popisky os*: Obsahuje textové popisky pro osy grafu, které pomáhají interpretovat zobrazená data.
  - *zobrazení pomocných čar*: Nabízí možnost zapnout nebo vypnout pomocné čáry na osách grafu, což usnadňuje čtení a interpretaci dat.
  - *velikost a ohraničení bodů*: Nastavuje velikost a styl bodů v bodovém grafu, což ovlivňuje jejich viditelnost a rozlišitelnost v grafu.
  - *styl čar*: Definuje styl čar použitých pro vykreslení grafu, například tloušťku a typ čáry.
  - *další specifické vlastnosti*: Další metadata a nastavení, která mohou být relevantní pro konkrétní typy grafů a potřeby uživatele.
- **tabulka**: Obsahuje pole sloupců, které reprezentují jednotlivé kategorie záznamů.

- *názvy sloupců*: Identifikují jednotlivé kategorie dat, které jsou zobrazovány v grafu.
- *barvy*: Určují barevné kódy, které jsou přiřazeny jednotlivým kategoriím dat pro vizualizaci.
- *data*: Obsahují samotná číselná data, která jsou zobrazena v grafu.

### 4.4.2 Rozšíření

I když současná funkcionalita aplikace nevyžaduje složitý databázový model, je vhodné připravit databázi i na budoucí rozšíření. To znamená, že je možné přidat další kolekce dat a vazby pro nové funkce nebo možnosti aplikace. Jeden příklad by mohl zahrnovat možnost sdílení grafů mezi uživateli, což by vyžadovalo novou tabulku pro sdílení a vazby mezi uživateli a grafy. Při použití relační databáze by bylo nutné upravovat tabulky a pečlivě plánovat budoucí změny. Díky tomu, že byla zvolena databáze MongoDB, která je určena pro ukládání částečně strukturovaných dat, tyto komplikace odpadají.

Je to také důvodem, proč byla pro nastavení grafu vytvořena položka metadata. Ne každý graf potřebuje všechna nastavení, kde příkladem může být koláčový graf, který nepotřebuje znát popisky os. Stávající i nová nastavení lze tedy ukládat pod atribut metadata a v případě přidávání nových funkcí je dynamicky měnit.

Kapitola 5

# Realizace

## 5.1 Struktura projektu

Projekt využívá výchozí adresářovou službu frameworku Symfony s několika drobnými změnami. Příkladem je změna složky Entity na složku Document, protože je použita dokumentová databáze narozdíl od výchozí relační databáze.

```
charts/
├── bin/ ..... binární soubory Symfony
├── config/ ..... konfigurační soubory balíčků a cest
├── public/
│   ├── index.php ..... vstupní soubor aplikace
│   └── scripts ..... JavaScriptové soubory
│       └── charts ..... knihovna pro vykreslování grafů
├── src/
│   ├── Api/
│   ├── Controller/
│   ├── Form/
│   ├── Repository/
│   │   └── DefaultController.php ..... výchozí controller
│   └── Document/ ..... MongoDB ekvivalent adresáře Entity
│       ├── User.php ..... entita uživatele
│       └── Chart.php ..... entita grafu
├── templates/
│   ├── base.html.twig ..... základní šablona
│   └── default/
│       └── index.html.twig ..... úvodní stránka
├── tests/
├── var/
├── vendor ..... balíčky závislostí
├── composer.json ..... definice závislostí
└── symfony.lock ..... zamknutí verzí balíčků
```

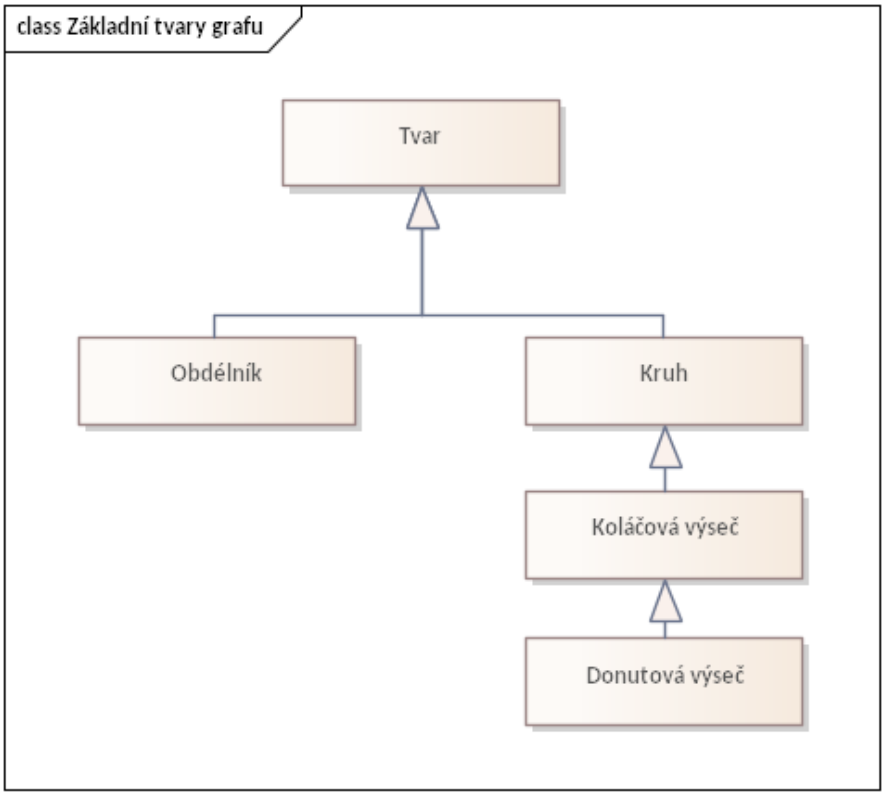
Obrázek 5.1: Adresářová struktura projektu

## 5.2 Vykreslování

Vykreslování grafu je rozděleno na několik hlavních částí a vrstev. První z nich je základní vrstva, která obsahuje osy, popisky data a další. Druhá vrstva se využívá k detekci interakcí a finální vrstva obsahuje efekty.

### 5.2.1 Základní tvary

Každý interaktivní element se nachází v první i druhé vrstvě a v případě interakce s ním se zobrazí i ve vrstvě třetí. Souřadnice a základní tvar bez ohraničení a stínů jsou pro každý prvek ve všech třech vrstvách totožné. V JavaScriptu je tedy uloženo pole všech interaktivních objektů, které reprezentují geometrické tvary v grafu. Zjednodušený diagram tříd je k vidění na obrázku 5.2.




Obrázek 5.2: Základní grafové tvary

- **Tvar**: Tvar představuje základní abstraktní entitu, která není přímo vykreslována, ale slouží jako počáteční bod pro další tvary v grafu. Jeho role spočívá v poskytování základních informací a parametrů, které budou děděny ostatními tvary. Obsahuje základní informace jako pozici a rozměry.
- **Obdélník**: Tvar obdélníku je odvozen z abstraktního tvaru. Jedná se o základní prvek, který představuje čtvercovou či obdélníkovou plochu

v grafu. Zahrnuje pozici, rozměry (šířku a výšku) a další charakteristiky tvaru.

- **Kruh**: Tvar kruhu vychází také z abstraktního tvaru. Je to základní prvek pro vykreslování kruhových grafů nebo bodů v grafu. Obsahuje informace o pozici středu, poloměru a další parametry definující jeho vzhled.
- **Koláčová výseč**: Tvar koláčové výseče je odvozen z tvaru kruhu. Reprezentuje část kruhového grafu, která je definována určitým úhlem. Zahrnuje pozici středu, poloměr, úhel a další parametry specifické pro koláčovou výseč.
- **Donutová výseč**: Donutová výseč je odvozením z koláčové výseče. Jedná se o vylepšenou verzi koláčové výseče, která obsahuje informace o vnitřním a vnějším poloměru, což jí dává prstencovitý tvar. Kromě běžných parametrů koláčové výseče obsahuje také informace o vnitřním a vnějším poloměru, které definují tloušťku prstence.

### 5.2.2 Základní třída grafu

Jedná se o základ, ze kterého všechny grafy vycházejí, a který obsahuje všechny společné funkce a atributy. Je zobrazena na diagramu 5.3. Ostatní grafy jsou v základu totožné a stačí proto uvést pouze tuto třídu.

#### 5.2.2.1 Atributy:

- **data: Array<Object>**: Tento atribut představuje pole objektů, které obsahují data určená k zobrazení na grafu. Každý objekt v poli představuje jednu sadu dat pro vykreslení.
- **settings: Object**: Atribut settings je objekt, který obsahuje různá nastavení pro graf, jako například barvu, styl čar nebo popisky os.
- **canvas: HTMLCanvasElement**: Tento atribut je odkaz na HTML element canvas, na kterém se bude kreslit graf. Canvas poskytuje prostředky pro vykreslování grafických prvků pomocí JavaScriptu.
- **zoom: ZoomManager**: Instance třídy ZoomManager, která umožňuje manipulaci se zoomem grafu. Pomocí této instance lze přiblížit nebo oddálit zobrazení grafu a provádět posouvání grafem.
- **ctx: CanvasRenderingContext2D**: 2D kreslící kontext canvasu, který umožňuje vykreslování grafických prvků na plátno. Tento kontext poskytuje metody pro kreslení tvarů, textu a dalších grafických prvků.
- **largest: number**: Největší hodnota v datech grafu. Tato hodnota představuje maximální hodnotu v celém datovém setu a je využívána pro určení měřítka grafu.
- **smallest: number**: Nejmenší hodnota v datech grafu. Tato hodnota představuje minimální hodnotu v celém datovém setu a je využívána pro určení měřítka grafu.


Obrázek 5.3: Diagram třídy základního grafu

- **dataLen: number**: Délka dat grafu. Tento atribut udává počet datových bodů, které budou zobrazeny na ose X grafu.
- **bounds: Object**: Objekt obsahující informace o hranicích grafu. Obsahuje informace o levém, pravém, horním a dolním okraji grafu a o pozici os.
- **zoomBounds: Object**: Objekt obsahující hranice grafu po aplikaci zoomu. Tento objekt poskytuje informace o hranicích grafu po přiblížení nebo oddálení zobrazení.
- **scale: number**: Měřítko grafu. Tento atribut určuje měřítko grafu na základě velikosti canvasu a rozsahu dat.

- **extreme: number**: Extrém grafu. Tento atribut určuje maximální absolutní hodnotu v celém datovém setu a je využíván při kreslení os.
- **objects: Array**: Pole objektů představujících tvary grafu. Tyto objekty představují jednotlivé prvky grafu, jako jsou body, čáry nebo oblasti.

#### 5.2.2.2 Funkce:

- **updateBounds()**: Aktualizuje hranice grafu na základě velikosti canvasu a nastavení odsazení. Tato metoda zajišťuje, aby graf správně vykreslil všechny prvky v rámci stanovených hranic.
- **getBounds(canvas: HTMLCanvasElement, graphMargin: number): Object**: Vypočítá hranice grafu na základě velikosti canvasu a okrajových hodnot. Tato metoda určuje rozsah hranic grafu na základě velikosti plátna a okrajových hodnot.
- **getZoomBounds(): Object**: Vrátí hranice grafu po aplikaci zoomu. Tato metoda získá hranice grafu po aplikaci zoomu na základě aktuálních hodnot přiblížení.
- **isInBounds(pos: Object): boolean**: Zjistí, zda je zadaná pozice uvnitř hranic grafu. Tato metoda určuje, zda je zadaná pozice uvnitř hranic grafu, což je užitečné pro detekci interakcí s grafem a zvyšuje rychlost vykreslování, protože se neplýtvá časem na zobrazení prvků, které není možné vidět.
- **drawTitle()**: Tato metoda vykresluje titulek grafu na dané pozici na plátně na základě zadaných hodnot o obsahu, fontu či velikosti písma.
- **updateLegend(displayLegend: boolean, legend: HTMLElement, chartLoader: any)**: Aktualizuje legendu na základě zadaných parametrů. Legenda obsahuje odkaz na každou kategorii dat.
- **resizeCanvas()**: Tato metoda upraví velikost plátna na základě velikosti jeho kontejneru.
- **clear()**: Vyčistí canvas a odstraní všechny uložené objekty.
- **drawDetectionMap(ctx: CanvasRenderingContext2D)**: Metoda vykreslí detekční mapu na daný 2D kontext, který se nachází na druhém canvasu. Využívá objektů a tvarů, které byly vyklesleny na základní vrstvě.
- **drawEffect(ctx: CanvasRenderingContext2D, objects: Array)**: Vykreslí efekty pro zadané tvary na daný 2D kontext.
- **drawAxis(displayAxisValues: boolean)**: Tato metoda vykresluje osy grafu na plátno a zároveň zobrazuje popisky os.
- **setClipRegion(x: number, y: number, w: number, h: number)**: Tato metoda nastaví ořezovou oblast pro daný kontext canvasu na základě zadaných parametrů. Tím se zabrání, aby vykreslovaná data zasahovala mimo určenou plochu.
- **drawYAxisTicks()**: Vykreslí značky a popisky na Y-ové ose.

- **drawXAxisTicks(displayAxisValues: boolean)**: Tato metoda vykreslí značky a popisky na X-ové ose grafu. Není využita pro sloupcové grafy, které popisky řeší jiným způsobem.
- **drawAxisLines()**: Tato metoda vykresluje čáry pro osy X a Y na plátno.
- **drawAxisLabels()**: Vykreslí popisky os grafu.

### 5.2.3 Zoom

Vzhledem k rastrové povaze canvasu není možné graf přiblížit bez ztráty kvality. Je proto nutné graf vždy po provedení zoomu překreslit. Zoom je implementován pomocí dvou oddělených systémů souřadnic. Jeden reprezentuje virtuální hranice grafu, které se během přibližování mohou nacházet i mimo canvas, zatímco druhý je využit přímo pro vykreslování na plátno. Graf lze přibližovat po horizontální i vertikální ose, přičemž tyto dvě jsou na sobě nezávislé.

### 5.2.4 Provedené optimalizace

Pro uživatele není příjemné, když aplikace provádí operace na pozadí a nepodává žádný vizuální obsah. Není z jejich pohledu možné určit, zda služba funguje správně, nebo zda došlo k chybě. Čím dříve se uživateli dostane vizuální zpětné vazby, tím lépe. Čas do prvního vykreslení je jednou z hlavních metrik při posuzování přívětivosti webových stránek. [39]

Jedná se o hlavní problém existujících řešení, které se na slabších zařízeních běžně vykreslují i několik vteřin. Toto řešení proto obsahuje mnoho různých optimalizací, které mají za cíl grafy vykreslit co nejrychleji.

#### 5.2.4.1 Asynchronní procesy

Během vývoje bylo implementováno asynchronní zpracování určitých operací, které nevyžadují okamžitou odpověď a mohou být vykonány v pozadí. To umožnilo efektivnější využití výpočetních prostředků a zlepšilo výkon aplikace. Konkrétně se jedná o načítání data pomocí AJAXu a jejich následné zpracování a vykreslení.

## 5.3 Backend

### 5.3.1 Dodatečné balíčky

Pro usnadnění práce v Symfony jsou použity dodatečné rozšiřující balíčky.

#### 5.3.1.1 Doctrine MongoDB Bundle

Doctrine dokáže v základu pracovat s relačními SQL databázemi. MongoDB je však dokumentová databáze, a tak vyžaduje dodatečný balíček, který zpracovává komunikaci mezi ní a Symfony. S tímto balíčkem se pracuje podobně jako standardní pro Doctrine, jen se místo Entity využívá Document a má pár rozdílných funkcí. [40]

#### 5.3.1.2 FOS REST Bundle

Symfony FOSRestBundle je rozšíření pro Symfony framework, které usnadňuje vytváření RESTful API v Symfony aplikacích. Poskytuje sadu nástrojů a funkcí pro rychlé a efektivní vytváření REST rozhraní včetně automatického mapování cest, serializace a deserializace dat, podporu pro formáty jako JSON a XML, řízení přístupu a mnoho dalšího. FOSRestBundle usnadňuje implementaci REST architektury ve Symfony aplikacích a umožňuje vývojářům snadno vytvářet robustní a dobře strukturovaná API. [41]

#### 5.3.1.3 Twig

Symfony Twig je šablonovací systém používaný v frameworku Symfony pro tvorbu dynamických uživatelských rozhraní ve webových aplikacích. Jedná se o moderní a výkonný nástroj, který umožňuje psát šablony ve formátu, který je čitelný pro člověka a zároveň poskytuje pokročilé funkce pro manipulaci s daty, podmíněné zobrazení, cykly a další. Twig usnadňuje oddělení logiky od prezentace a poskytuje strukturovaný a elegantní způsob tvorby uživatelských rozhraní v Symfony aplikacích. [42]

### 5.3.2 API

API pro správu grafů poskytuje několik koncových bodů pro získání seznamu grafů, zobrazení detailů konkrétního grafu, vkládání nových grafů, aktualizaci stávajících grafů a mazání grafů.

- **GET /api/charts**: Vrací seznam ID grafů, jež patří přihlášenému uživateli. Odpověď obsahuje seznam ID grafů a HTTP kód 200 OK.
- **GET /api/charts/{id}**: Endpoint vrací detailní informace o grafu na základě jeho ID. Odpověď obsahuje detailní informace o grafu a kód 200.
- **POST /api/charts/insert**: Umožňuje uživateli vytvořit nový graf. Data grafu se posílají v těle zprávy ve formátu JSON. Backend následně vytvoří novou entitu a přiřadí jí unikátní ID. Odpověď obsahuje název vytvořeného grafu s HTTP kódem 200. Uživatel musí být přihlášen, jinak se vrátí chyba přístupu s HTTP kódem 403.
- **POST /api/charts/{id}/update**: Umožňuje uživateli aktualizovat jeho grafy. Data se posílají v těle zprávy jako JSON. Obsahem odpovědi je název aktualizovaného grafu s HTTP kódem 200. Pokud graf patří jinému uživateli, vrátí se chyba přístupu s HTTP kódem 403.
- **DELETE /api/charts/{id}/remove**: Umožňuje smazat existující graf. Odpověď obsahuje název smazaného grafu s HTTP kódem 200. Uživatel může mazat pouze své vlastní grafy, jinak se vrátí chyba přístupu s kódem 403.

## 5.4 Nasazení

Pro nasazení aplikace na server je nezbytné zvolit vhodnou infrastrukturu a správně nakonfigurovat prostředí pro běh aplikace. V tomto případě je apli-

kace nasazena na virtuálních počítačích, které poskytují potřebné výpočetní prostředky a flexibilitu pro škálování podle potřeby.

Tyto virtuální počítače běží na clusteru tří fyzických zařízení. Jako hypervizor a správce clusteru je využit Proxmox, který umožňuje centralizovanou správu virtuálních strojů a kontejnerů a poskytuje prostředky jako jsou možnosti zálohování, migrace a škálování zdrojů.

Aplikace však může běžet i na jednom počítači, kde webserver a databáze běží přímo v operačním systému nebo v docker kontejnerech.

### 5.4.1 Webový Server

Pro přijímání a zpracování HTTP požadavků od klientů je použit webový server Apache2, který je jedním z nejpopulárnějších webových serverů díky své spolehlivosti, široké podpoře a flexibilitě. Apache2 je široce využíván ve webovém prostředí díky své schopnosti snadno integrovat další moduly a rozšíření pro různé potřeby. Jeho konfigurace umožňuje nastavení různých funkcí, jako je řízení přístupu, logování událostí a podpora dynamického zpracování obsahu.

### 5.4.2 Databázový Systém

Pro ukládání a správu dat je použit databázový systém MongoDB. Ten běží na dalším virtuálním počítači odděleném od webového serveru. Tím se zajišťuje izolace dat a minimalizuje se riziko jejich ztráty nebo poškození. Je možné vyměnit databázový server nebo zprovoznit další instance, které mohou také přistupovat k této databázi. Pro zvýšenou bezpečnost dat se ukládají na redundantní pole disků, které je odolné proti selhání až dvou disků najednou.

Kapitola 6

# Testování

Tématem této kapitoly je otestování vytvořeného řešení, což je klíčový krok při zajišťování kvality a funkčnosti aplikace. Proces testování lze rozdělit do dvou hlavních kategorií. První kategorií je technická část, která se zaměřuje na měření a hodnocení výkonnosti aplikace. To zahrnuje analýzu nároků na výkon, jako je rychlost načítání, odezva uživatelského rozhraní a efektivita paměťového využití. Druhou kategorií je testování řešení skutečnými uživateli, což je nezbytný krok pro zajištění uživatelské spokojenosti a použitelnosti aplikace.

## 6.1 Testování nároků na výkon

Jedním z hlavních cílů práce bylo vytvořit řešení, které je na rozdíl od dosavadních služeb rychlé a vhodné i pro slabší zařízení. Z tohoto důvodu nestačí otestovat pouze tuto práci, ale je nutné změřit i ostatní služby, aby bylo možné porovnat, zda cíle byly skutečně splněny. Pro tento účel byly zvoleny řešení uvedené v kapitole o analýze 2.1. Byla však vynechána služba LiveGap Charts, protože v základní verzi generuje pouze statické grafy, což neodpovídá potřebám tohoto testování.

### 6.1.1 Testovací prostředí a metodika

Aby naměřené hodnoty bylo možné vzájemně porovnávat a vyvozovat z nich závěry, tak musí všechno testování probíhat ve stejném pevně určeném prostředí. Zvoleným zařízením je notebook od značky HP s procesorem i5-8265U a šestnácti gigabajty RAM[23]. Jedná se o zařízení s průměrným výkonem a díky tomu by naměřené hodnoty měly odpovídat tomu, s čím se uživatelé setkávají nejčastěji. Zvoleným webovým prohlížečem je Mozilla Firefox, s nímž je autor práce zvyklý pracovat. Nejvíce používaným prohlížečem je sice Google Chrome, ale během základního otestování mezi různými prohlížeči nebyly znatelné rozdíly při vykreslování grafů, a tak je možné zvolit kterýkoliv z nich.

Každá služba má své vlastní, mírně odlišné použití. Některé se specializují na vizuální stránku a efekty, zatímco jiné jsou určené pro podrobné datasety. Na první pohled je také jasné, že například koláčový graf se vykresluje úplně jinak než spojnicový. Pro zachování konzistence v testování byl zvolen sloupcový

[23] Random Access Memory

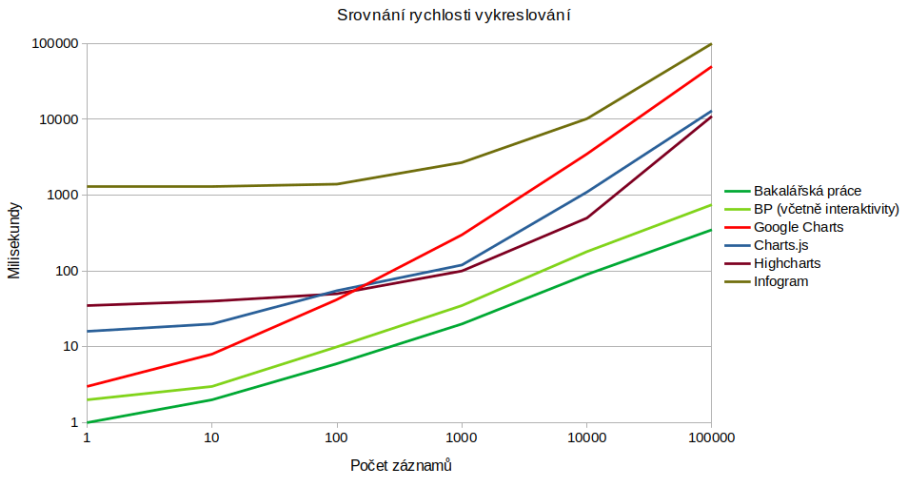
graf a pro každou službu bylo použito doporučené nastavení, které je uvedeno v dokumentaci. Každý graf byl implementován na samostatné webové stránce, aby se minimalizoval vliv dalšího obsahu na výsledky testů. Rozsah dat pro vykreslení byl od jednoho až po sto tisíc záznamů, přičemž hodnoty byly náhodně generované v rozsahu nula až tisíc.

Pro získání hodnot byly využity funkce JavaScriptu pro měření času a nástroje pro vývojáře, které jsou součástí webového prohlížeče. Ty zahrnují analýzu síťového provozu, prohlíženi struktury DOM[24], kategorizování a změření obsahu RAM a další. Každé měření bylo provedeno několikrát a nakonec z nich byla vypočítána průměrná hodnota, aby se snížily výkyvy a zvýšila se přesnost měření.

### 6.1.2 Rychlost vykreslování grafů

Tato sekce se zaměřuje na měření rychlosti vykreslování grafů u jednotlivých služeb. Jedná se o jeden z primárních faktorů, které mají veliký dopad na uživatelský zážitek. [43] Vzhledem k tomu, že každá služba využívá jiný způsob načítání dat a rychlost internetu nebo úložiště se může lišit, tak bylo měřeno pouze samotné vykreslování grafu. Naměřené hodnoty tedy neobsahují informace o načítání dat.

Na obrázku 6.1 je vidět graf naměřených hodnot. Vzhledem k rozsahu testovaných dat byl pro obě osy grafu zvolena logaritmická stupnice, která umožňuje přehledně zobrazit nízké i velmi vysoké hodnoty. Nižší hodnoty značí v tomto grafu lepší výsledek.




Obrázek 6.1: Srovnání rychlosti vykreslování grafů pomocí různých služeb

Tato práce se na grafu nachází rovnou dvakrát, protože je její vykreslování rozděleno na dvě fáze. Nejprve se pouze vykreslí informace na základní vrstvu grafu, aby je uživatel mohl vidět co nejdříve po vstupu na webovou stránku. Jedná se však pouze o statický graf. Ve druhé fázi se vykreslí detekční vrstva

[24] Document Object Model

a připraví se interaktivní prvky. Na obrázku je tedy vidět čas do prvního zobrazení grafu a také čas do plného načtení a zprovoznění všech funkcí.

#### 6.1.2.1 Infogram

Nejhůře je na tom služba Infogram, která je i pro malé množství dat znatelně pomalejší. Sto hodnot se vykresluje téměř stejně dlouho jako pouze jedna hodnota. Je to dáno tím, že během vykreslování grafu stahuje mnoho dodatečných informací z internetu a způsobuje tím dodatečné zdržení, které není závislé na vykreslovaných hodnotách. Následně také generuje mnoho HTML elementů, které slouží k efektům nebo ke sdílení na sociální sítě, některé z nich však nemají vůbec žádný účel. [4]

Výborně se tím ilustrují problémy moderních služeb, které se soustředí na marketing a nové funkce, přičemž však zapomínají na optimalizaci a ve výsledku tím zhoršují uživatelský zážitek. [44]

#### 6.1.2.2 Google Charts

Dále přichází na řadu Google Charts. Na obrázku je patrné, že Google Charts exceluje při práci s menšími datasety, typicky do přibližně sta hodnot. Jeho rychlost a efektivita jsou v tomto rozsahu velmi solidní. Nicméně, jakmile se překročí určitá hranice a pracuje se s větším množstvím dat, tak jej začnou překonávat jiné služby. Je tedy vhodné pro jednoduché vizualizace, ale nehodí se pro podrobnější a rozsáhlejší grafy. [1]

#### 6.1.2.3 Charts.js

Charts.js je kromě této práce jediným reprezentantem canvasu. Oproti ostatním, které využívají SVG, má tedy zásadní výhodu v rychlosti vykreslování. Tato vlastnost mu umožňuje efektivně pracovat s velkým objemem dat a zajistit plynulou vizualizaci i při zpracování rozsáhlejších datových sad. Na obrázku 6.1 je patrné, že Charts.js exceluje zejména při práci s většími datasety, kde je několikrát rychlejší než konkurenční služby, jako je Infogram a Google Charts.

#### 6.1.2.4 Highcharts

Highcharts má podobnou rychlost jako Charts.js, přestože používá pomalejší technologie. Jedná se o placenou službu, která se zaměřuje na rozsáhlé datasety a je tedy pro tento účel optimalizovaná. Naměřený výsledek odpovídá očekávaným hodnotám a potvrzuje, že je pro svůj účel vhodnou volbou.[2]

#### 6.1.2.5 Nové řešení

Nakonec se na grafu nachází nové řešení vytvořené v rámci této práce. Je nejrychlejší pro všechny rozsahy dat, od pouhých jednotek záznamů až po statisíce. Jako jediná ze služeb nacházejících se v grafu 6.1 byla tato aplikace otestována i pro miliony záznamů, které zvládá vykreslit během pouhých několika vteřin. Ostatní služby nedokázaly takový dataset vůbec vykreslit, protože je limitoval zvolený hardware. V praxi se samozřejmě nevytváří grafy s takovým množstvím dat, je to však dobrou ilustrací rychlosti a efektivity tohoto řešení. Zavedení

interaktivity trvá přibližně stejný čas jako prvotní vykreslení. I tak je však výrazně rychlejší než ostatní služby.

Zjištěné výsledky přinesly příjemné překvapení. Bylo možné očekávat, že toto nové řešení bude pro větší množství dat rychlejší než ostatní služby, protože se jedná o jeden z hlavních cílů této práce a bylo mu věnované značné úsilí. U menších dat se nečekalo významné zlepšení, protože se optimalizace na menších datasetech nestačí projevit. Ukázalo se však, že nové řešení dokáže významně zrychlit vykreslování i menšího množství dat. U větších datasetů se rychlost zvyšuje řádově výše, než bylo očekáváno – namísto jednociferného násobku se rychlost zvýšila dokonce desetkrát až stokrát.

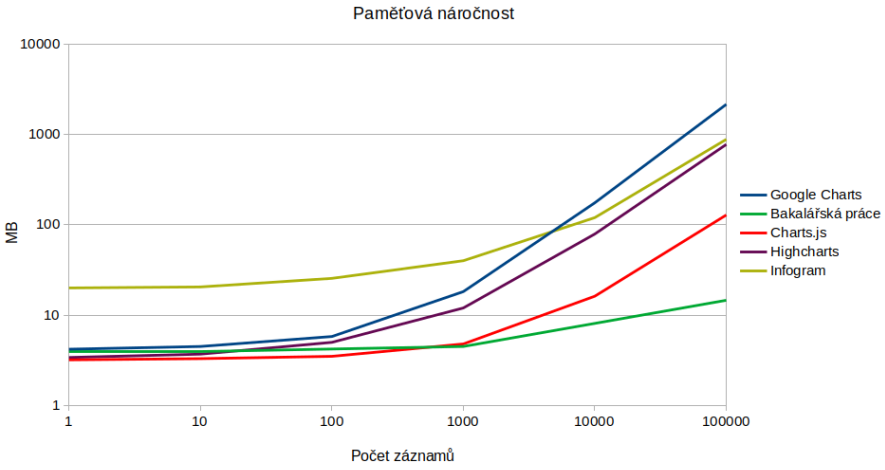
### 6.1.3 Paměťová náročnost

Tato sekce se zaměřuje na analýzu paměťové náročnosti, která je dalším důležitým faktorem webových aplikací. Často se totiž využívají i na slabších zařízeních, jako jsou starší počítače či mobilní telefony, které mají omezené hardwarové prostředky.

Naměřené hodnoty jsou k vidění na obrázku 6.3. Byly zde zvoleny stejné služby jako v předchozí sekci. Jediným rozdílem je, že se tato práce na grafu nachází pouze jednou. Není totiž potřeba měřit jednotlivé fáze vykreslování, protože významná je pouze celková paměťová náročnost. Stejně jako tomu bylo u testování rychlosti, je i zde použita logaritmická stupnice a nižší hodnoty značí méně vyžadované RAM a výsledek je tedy lepší.


Obrázek 6.2: Analýza použité poměti

Na obrázku lze vidět, že při menším množství dat hrají hlavní roli ve spotřebě paměti ostatní prvky, jako jsou načtené skripty, základní struktura webové stránky, CSS soubory a podobně. Při větších datasetech se však začnou projevovat i samotná data, a to ve formě HTML elementů a ve formě uložených objektů v rámci JavaScriptu. Pro zjištění konkrétních hodnot byla využita funkce analýzy paměti, která je součástí nástrojů pro vývojáře ve Firefoxu – výsledné diagramy vypadají jako obrázek 6.2.




Obrázek 6.3: Srovnání spotřeby operační paměti

#### 6.1.3.1 Infogram

Nejhůře ze všech aplikací je na tom opět Infogram, který nehledě na množství dat načítá mnoho dodatečných prvků. [4] Znamená to, že vyžaduje zbytečně mnoho paměti i pro malé datasety, což může negativně ovlivnit uživatelský zážitek a výkon aplikace. Při větším množství dat však není tato chyba tak znatelná a výsledky se blíží ostatním podobným službám.

#### 6.1.3.2 Google Charts

Google Charts vyžaduje nízké množství paměti pro malé datasety, ale při větším množství dat jeho spotřeba rychle stoupá. Na grafu je vidět, že okolo pěti set záznamů dokonce spotřebovává více RAM než Infogram. Toto chování se podobá tomu, co bylo naměřeno v předchozí sekci. Služba je tedy efektivní pouze pro menší množství dat.

#### 6.1.3.3 Highcharts

Highcharts má ze všech služeb, které používají technologii SVG, nejnižší nároky na operační paměť. Využívá méně HTML elementů než obě předchozí služby a dosahuje tak větší efektivity.

#### 6.1.3.4 Charts.js

Zatím byly zmíněny pouze grafy vykreslované pomocí SVG, které má mnohem větší nároky na paměť než canvas. Nyní však přichází na řadu služba Charts.js. Na obrázku 6.3 je možné vidět, že má řádově nižší nároky na operační paměť. To je způsobeno tím, že canvas má vždy víceméně konstantní velikost, protože se v podstatě jedná o rastrový obrázek. Vektorová grafika musí naopak ukládat každý objekt ve své stromové struktuře, a tak se její velikost zvětšuje s rostoucím počtem vykreslovaných tvarů a dalších prvků. [22] Jediné co spotřebovává

operační paměť jsou tedy objekty v JavaScriptu, které se využívají pro zajištění interaktivity.

#### 6.1.3.5 Nové řešení

Nakonec se dostává na řadu toto nové řešení. Na grafu je znázorněno, že pro malé množství dat má srovnatelnou spotřebu paměti jako jiná řešení, dokonce mírně horší. Důvodem je pravděpodobně to, že jsou pro vykreslení grafu použity tři canvasy, které zabírají určité množství RAM bez ohledu na množství zobrazovaných dat. Pro větší datasety je však několikanásobně lepší než všechny ostatní služby. Bylo potřeba pouhých 14 megabajtů pro to samé, na co Google Charts spotřebovalo přes dva gigabajty. Společně s Charts.js se jednalo o jediná řešení, která dokázala zobrazit milion hodnot. Ostatní služby toho nebyly na zvoleném hardwaru vůbec schopné. Překvapivým zjištěním bylo, že toto řešení využívá řádově méně RAM než Charts.js, přestože se obě zakládají na stejné technologii.

## 6.2 Testování s uživateli

Tato podkapitola se zabývá testováním designu aplikace a uživatelské přívětivosti. Rychlost a nároky na operační paměť totiž nejsou jedinými faktory, které mají vliv na uživatelský zážitek.

### 6.2.1 Registrace a přihlášení

Registrace a přihlášení jsou klíčové části uživatelského zážitku, které ovlivňují celkovou uživatelskou přívětivost aplikace. V této části je testován proces registrace nového uživatele a přihlášení do aplikace, což jsou první kroky, které uživatelé musí provést při interakci s aplikací.

1. **Chybné údaje:**
    - Uživatel má za úkol zkusit vyplnit registrační formulář s neplatnými údaji a otestovat, jak aplikace zpracuje a zobrazí chyby.
2. **Vyplnění registračního formuláře:**
    - Úkolem uživatele je vytvoření nového uživatelského účtu pomocí registračního formuláře.
3. **Úspěšné přihlášení:**
    - Cílem tohoto úkolu je přihlášení do aplikace pomocí uživatelových přihlašovacích údajů, které vyplnil v předchozím kroku.

### 6.2.2 Vytvoření nového grafu

Tato sekce testování se zaměřuje na proces vytváření nových grafů, což je hlavním účelem této aplikace, přičemž je sledována uživatelská schopnost snadno se orientovat a pracovat s grafickými nástroji. Cílem je zajistit, aby uživatelé mohli efektivně vytvářet a upravovat grafy podle svých potřeb a požadavků.

#### 6.2.2.1 Vytvoření grafu:

1. **Založení nového grafu:**
   - Účelem tohoto úkolu je zjistit, zda se uživatel dokáže po přihlášení v aplikaci zorientovat a vytvořit nový graf.

#### 6.2.2.2 Vložení dat:

Cílem této části testování je ověřit, zda je proces vkládání dat snadno proveditelný, přehledný a umožňuje uživateli efektivně pracovat s daty.

1. **Import dat ze souboru:**
   - Uživatel má za úkol naimportovat data do grafu z externího souboru, jako je CSV.
2. **Úprava dat pomocí GUI:**
   - Uživatel má za úkol změnit některé hodnoty pomocí tabulkového editoru ve webové aplikaci.
3. **Přidání nového sloupce:**
   - Úkolem uživatele je vložení nového sloupce do tabulky a naplnění daty včetně nadpisu.

#### 6.2.2.3 Úprava nového grafu:

1. **Změna pozadí:**
   - Uživatel má za úkol změnit typ a styl grafu v průběhu jeho tvorby podle požadavků.
2. **Přidání nadpisu:**
   - Úkolem uživatele je přidání hlavního nadpisu grafu a nastavení jeho fontu a velikosti písma.
3. **Zapnutí legendy:**
   - Uživatel má možnost zapnout legendu, která pomáhá identifikovat jednotlivé kategorie dat v grafu a jejich význam.
4. **Změna typu grafu:**
   - Uživatel má za úkol změnit typ grafu podle svého uvážení. Výchozí graf je nastaven jako spojnicový, nový typ může tedy být bodový, sloupcový, koláčový a podobně.
5. **Změna barvy sloupce:**
   - Úkolem uživatele je změna barvy jedné kategorie dat, kterou reprezentuje sloupec v tabulce. Toto nastavení má vliv na barvu bodů, sloupců či legendy.

### 6.2.3 Export dat

Tato část testování se zaměřuje na proces exportu dat z aplikace, což je důležitá funkcionalita pro uživatele při sdílení dat s ostatními nebo pro uchování dat na další zpracování.

1. **Export dat do formátu CSV:**
   - Při tomto úkolu se testuje schopnost uživatele exportovat data z aplikace do formátu CSV pro další zpracování v externích nástrojích.
2. **Export grafu do obrázku:**
   - Tento úkol ověřuje, zda uživatel dokáže exportovat aktuální graf do formátu obrázku (např. PNG nebo JPEG) pro sdílení s ostatními.

#### 6.2.3.1 Interakce s grafem:

Tato část testování se zaměřuje na interakci uživatele s grafem a jeho prvky. Zahrnuje různé funkce, které umožňují uživateli lépe porozumět datům a efektivněji pracovat s grafickými vizualizacemi.

1. **Zobrazení detailu:**
   - Uživatel má za úkol přiblížit se k určité části grafu a zobrazit detailní informace o konkrétních datech nebo hodnotách. Ověřuje se tak možnost zobrazení tooltipu (bubliny vedle kurzoru) s podrobnostmi.
2. **Označení kategorie dat pomocí legendy:**
   - Během tohoto úkolu se uživatel pokusí označit konkrétní kategorie dat v grafu pomocí legendy a zhodnotit, jak je tato funkce přínosná pro přehlednost grafu a prezentaci dat.
3. **Zoomování na graf:**
   - Uživatel má za úkol přibližovat a oddalovat graf, aby lépe prozkoumal jednotlivé části dat. Testuje se plynulost a snadnost zoomování a možnost navigace po grafu.

### 6.2.4 Výsledky

Během testování bylo objeveno několik technických chyb. Příkladem je odsazení grafu, které při vysokých hodnotách blokovalo vykreslování sloupcových grafů, nebo že se webová aplikace zasekla, když se v tabulce nacházelo příliš mnoho dat. Dalším příkladem je ořízntutí obsahu na menších obrazovkách či nefunkčnost některých vlastností při použití dotykové obrazovky. Všechny chyby odhalené během testování byly vzápětí opraveny.

Grafický design se uživatelům zamlouval a příznivě hodnotili také rychlost vykreslování grafů a jejich intuitivní ovládání.

# Závěr

V rámci této bakalářské práce byla vyvinuta přívětivá webová aplikace pro tvorbu a úpravu interaktivních grafů. Postupováno bylo podle stanovených cílů, které byly definovány v úvodní kapitole. Hlavním cílem bylo poskytnout uživatelům prostředí pro snadnou tvorbu a úpravu grafů s využitím moderních technologií. Mezi dílčí cíle bylo provedeno provedení analýzy existujících řešení, návrh architektury aplikace, implementace klíčových funkcí a testování celého systému.

Nejdříve byla provedena analýza existujících nástrojů a služeb pro tvorbu grafů a technologií vhodných pro tuto aplikaci. Poté byly definovány požadavky na aplikaci a rozděleny do různých kategorií podle jejich důležitosti a priority. Na základě těchto požadavků byly zvoleny vhodné technologie a navržena architektura a uživatelské rozhraní aplikace.

Během implementační fáze byl kladen důraz na vytvoření klíčových funkcí, jako je tabulkový editor pro zadávání dat, nastavení vzhledu grafu a vizualizace výsledného grafu. Také byla provedena implementace interaktivních prvků a optimalizace výkonu aplikace.

Výsledná aplikace obsahuje přehledné uživatelské rozhraní, které umožňuje snadnou tvorbu a úpravu grafů. Uživatelé mohou využívat širokou škálu funkcí pro vytváření a personalizaci grafů podle svých potřeb. Během testování rychlosti vykreslování a paměťové náročnosti dokonce toto řešení předčilo očekávání ve třech důležitých aspektech. Těmi jsou rychlost vykreslování grafů u malých dat, řádově vyšší rychlost vykreslování u větších datasetů oproti ostatním službám a výrazně nižší nároky na operační paměť než u ostatních aplikací. Testování s uživateli poté pomohlo doladit prostředí aplikace. Všechny stanovené cíle byly splněny a aplikace je připravena k dalšímu rozvoji a rozšíření o další funkce a vylepšení.

# Literatura


[1] Charts | Google for Developers. [online] https://developers.google.com/chart, September 23 2023, [cit. 01-03-2024].

[2] Highcharts Documentation. [online] https://www.highcharts.com/docs, 2024, [cit. 01-03-2024].

[3] Chart.js. [online] https://www.chartjs.org/, [cit. 01-03-2024].

[4] Create Infographics, Reports and Maps – Infogram. [online] https://infogram.com/, [cit. 01-03-2024].

[5] Sedki, O.: Online Chart & Graph Maker| LiveGap. [online] https://charts.livegap.com/, 2024, [cit. 21-03-2024].

[6] Creative, A.: The Differences Between Illustrator and InDesign. [online] https://www.ashworthcreative.com/blog/2014/07/adobe-indesign-adobe-illustrator-differ/, Červenec 2014, [cit. 15-04-2024].

[7] Craig, W.: 10 Useful Flash Components for Graphing Data. [online] https://www.webfx.com/blog/web-design/10-useful-flash-components-for-graphing-data/, Duben 2009, [cit. 15-04-2024].

[8] Finley, D. R.: Determining Whether A Point Is Inside A Complex Polygon. [online] https://alienryderflex.com/polygon/, [2002], [Accessed 24-04-2024].

[9] Kuan, J.: *Learning Highcharts*. Birmingham, England: Packt Publishing, Prosinec 2012, ISBN 978-1849519083.

[10] Fhala, B.: *HTML5 graphing and data visualization cookbook*. Birmingham, England: Packt Publishing, Listopad 2012, ISBN 9781849693714.

[11] Microsoft: Prezentace dat v bodovém nebo spojnicovém grafu. [online] https://support.microsoft.com/cs-cz/office/prezentace-dat-v-bodov%C3%A9m-nebo-spojnicov%C3%A9m-grafu-4570a80f-599a-4d6b-a155-104a9018b86e, (2024), [Accessed 11-05-2024].

[12] Microsoft: Available chart types in Office. [online] https://support.microsoft.com/en-us/office/available-chart-types-in-office-a6187218-807e-4103-9e0a-27cdb19afb90, (2020), [Accessed 11-05-2024].

[13] Systems, A.: What are raster image files? | Adobe. [online] https://www.adobe.com/creativecloud/file-types/image/raster.html, [cit. 19-03-2024].

[14] Systems, A.: Vector files: How to create, edit and open them | Adobe. [online] https://www.adobe.com/creativecloud/file-types/image/vector.html, [cit. 19-03-2024].

[15] Kilin, I.: To click or not to click: static vs. interactive charts. [online] https://www.datylon.com/blog/pros-and-cons-of-static-and-interactive-charts, Březen 2023, [cit. 14-04-2024].

[16] Raggett, D.: Objects, Images, and Applets in HTML documents. [online] https://www.w3.org/TR/html401/struct/objects.html, Březen 2018, [cit. 11-04-2024].

[17] Schiller, T.: A Brief History of Browser Extensibility. [online] https://medium.com/brick-by-brick/a-brief-history-of-browser-extensibility-bcfeb4181c9a, Březen 2021, [cit. 14-04-2024].

[18] Root, E.: The history of Flash. [online] https://www.kaspersky.com/blog/life-and-death-of-adobe-flash/45906/, Říjen 2022, [cit. 14-04-2024].

[19] Morkes, D.: Java Applet krok za krokem. [online] https://www.interval.cz/clanky/java-applet-krok-za-krokem/, Září 2002, [cit. 15-04-2024].

[20] Byrne, M.: The Rise and Fall of the Java Applet. [online] https://www.vice.com/en/article/8q8n3k/a-brief-history-of-the-java-applet, Únor 2016, [cit. 15-04-2024].

[21] Fulton, S.: *HTML5 Canvas*. Sebastopol, CA: O'Reilly Media, druhé vydání, Květen 2013, ISBN 978-1-449-33498-7.

[22] Foundation, M.: SVG Tutorial – SVG: Scalable Vector Graphics. [online] https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial, Březen 2023, [cit. 16-04-2024].

[23] Foundation, M.: SVG API – Web APIs. [online] https://developer.mozilla.org/en-US/docs/Web/API/SVG_API, Prosinec 2023, [Accessed 23-04-2024].

[24] Foundation, M.: HTML: HyperText Markup Language. [online] https://developer.mozilla.org/en-US/docs/Web/HTML, Březen 2024, [cit. 20-03-2024].

[25] Lie, H. W.; Bos, B.: *Cascading style sheets:Designing for the web*. Boston, MA: Addison Wesley, druhé vydání, Červenec 1999, ISBN 0-201-59625-3.

[26] Achour, M.: JavaScript | MDN. [online] https://developer.mozilla.org/en-US/docs/Web/JavaScript, [cit. 8-03-2024].

[27] PHP: PHP Manual. [online] https://www.php.net/manual/en/, March 17 2024, [cit. 19-03-2024].

[28] Project, S.: Symfony, High Performance PHP Framework for Web Development. [online] https://symfony.com/at-a-glance, 2024, [cit. 21-03-2024].

[29] documentation, J.: Java Documentation – Get Started. [online] https://docs.oracle.com/en/java/, [cit. 20-03-2024].

[30] Nourie, D.: Java Technologies for Web Applications. [online] https://www.oracle.com/technical-resources/articles/java/webapps.html, Listopad 2006, [cit. 21-03-2024].

[31] SpringFramework: Spring Framework. [online] https://spring.io/projects/spring-framework, [cit. 20-03-2024].

[32] Nodejs: Node.js – About Node.js®. [online] https://nodejs.org/en/about, 2024, [cit. 21-03-2024].

[33] Karrys, L.: About npm | npm Docs. [online] https://docs.npmjs.com/about-npm, Říjen 2023, [cit. 15-04-2024].

[34] MySQL: MySQL :: MySQL 8.3 Reference Manual :: 1.2.1 What is MySQL? [online] https://dev.mysql.com/doc/refman/8.3/en/what-is-mysql.html, Prosinec 2023, [cit. 21-03-2024].

[35] PostgreSQL: PostgreSQL: Documentation [online]. [online] https://www.postgresql.org/docs, [cit. 19-03-2024].

[36] MongoDB: MongoDB Manual. [online] https://www.mongodb.com/docs/manual, [cit. 19-03-2024].

[37] Smus, B.: Improving HTML5 Canvas performance. [online] https://web.dev/articles/canvas-performance, Srpen 2011, [Accessed 23-04-2024].

[38] Koďousková, B.: Co je to API a jaké jsou možnosti jeho využití? [online] https://www.rascasone.com/cs/blog/co-je-api, Duben 2024, [Accessed 14-05-2024].

[39] Schapira, B.: First Contentful Paint (FCP), Start Render, First Paint. How to properly measure the beginning of page rendering? [online] https://blog.dareboost.com/en/2019/09/first-contentful-paint-fcp/, Září 2019, [Accessed 16-05-2024].

[40] Project, D.: DoctrineMongoDBBundle. [online] https://www.doctrine-project.org/projects/doctrine-mongodb-bundle/en/5.0/index.html, Leden 2024, [Accessed 22-04-2024].

[41] a další, L. K. S.: FOSRestBundle. [online] https://github.com/FriendsOfSymfony/FOSRestBundle, 2024, [Accessed 14-05-2024].

[42] Project, S.: Documentation – Twig – The flexible, fast, and secure PHP template engine. [online] https://twig.symfony.com/doc/, [2020], [Accessed 14-05-2024].

[43] Buruk, B.: Mastering Web Application Performance: A Comprehensive Guide for Quick Wins and Lasting Improvements. [online] https://medium.com/@burak.bburuk/supercharge-your-web-application-unleashing-the-hidden-tricks-to-boost-performance-c66ac0e265bd, Červenec 2023, [Accessed 13-05-2024].

[44] Prince, S.: How Poor Website Design Could Affect Your Performance. [online] https://www.clickthrough.digital/poor-website-design-affecting-performance/, Září 2023, [Accessed 13-05-2024].

Příloha A

# Seznam použitých zkratek

**SVG** Scalable Vector Graphics

**GIF** Graphics Interchange Format

**REST** Representational State Transfer

**API** Application Programming Interface

**XML** Extensible Markup Language

**CSV** Comma Separated Values

**XLS** Excel spreadsheet

**JPEG** Joint Photographic Experts Group

**PNG** Portable Network Graphics

**HTML** Hypertext Markup Language

**CSS** Cascading Style Sheets

**W3C** World Wide Web Consortium

**JIT** Jist-in-time

**AJAX** Asynchronous JavaScript and XML

**OOP** Objektově Orientované Programování

**JVM** Java Virtual Machine

**JSP** JavaServer Pages

**SQL** Structural Query Language

**JSON** JavaScript Object Notattion

**ACID** Atomicity, Consistency, Isolation, Durability

**GIMP** GNU Image Manipulation Program

**RAM** Random Access Memory

**DOM** Document Object Model

**GUI** Graphical User Interface

Příloha B

# Obsah příloh

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF