Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Moderní Javascriptové frameworky

Bc. Martin Dulovec

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Martin Dulovec

Systémové inženýrství a informatika
Informatika

Název práce

Moderní Javascriptové frameworky

Název anglicky

Modern Javascript Frameworks

Cíle práce

Práce bude tématicky zaměřena na problematiku moderních Javascriptových(JS) frameworků. Hlavním cílem diplomové práce je zanalyzovat a vyhodnotit vybrané JavaScriptové frameworky pro vývoji bussiness aplikací.
Dílčí cíle jsou:
- charakterizovat problematiku vývoje webových aplikací,
- analyzovat vybrané Javascriptové Frameworky
- formulovat doporučení k nasazení Javascriptových frameworků pro vývoj bussiness a webových aplikací

Metodika

Teoretická část diplomové práce se bude zaměřovat na literární přehled odborných zdrojů, zaměřených na problematiku vývoje aplikací pomocí moderních Javasriptových frameworků.

V praktické části budou vybrány konkrétní Javascriptové frameworky, které budou zhodnoceny dle zvolených kritérií. Na základě výsledků budou formulovány doporučení pro jejich využití k vývoji bussines a webových aplikací.

Na základě teoretické a praktické části dojde ke zformulování získaných poznatků a budou zpracovány závěry diplomové práce

Doporučený rozsah práce

60–80 stran

Klíčová slova

Javascript, Angular, React, Vue, Web development, Single Page

---

Doporučené zdroje informací

DOMES, M. *Tvorba internetových stránek pomocí HTML, CSS a JavaScriptu.* Kralice: Computer Media, 2005. ISBN 80-86686-39-6.

FLANAGAN, D. *JavaScript : the definitive guide.* Sebastopol, CA: O'Reilly, 2002. ISBN 0-596-00048-0.

HOLZNER, S. *JavaScript : profesionálně : [kompletní referenční příručka].* Praha: Mobil Media, 2003. ISBN 80-86593-40-1.

HOQUE, S. *Full-Stack React Projects : Modern Web Development Using React 16, Node, Express, and MongoDB. [elektronický zdroj] /.* Birmingham: Packt Publishing, Limited, 2018. ISBN 9781788832946.

POLLOCK, J. *JavaScript : příručka programátora.* Praha: Softpress, 2003. ISBN 80-86497-44-5.

Předběžný termín obhajoby

2022/23 ZS – PEF

Vedoucí práce

Ing. Michal Stočes, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 14. 7. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 28. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 28. 11. 2022

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Moderní Javascriptové frameworky" jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne datum odevzdání _____

Poděkování

Rád bych touto cestou poděkoval Ing. Michalu Stočesovi, Ph.D. za rady, připomínky a metodické vedení této práce.

Moderní Javascriptové frameworky

Abstrakt

Diplomová práce řeší problematiku zhodnocení a analýzu JS frameworků. Teoretická část řeší oblasti vztahující se k praktické části jsou to zejména funkcionality JS frameworků a jejich nadstavbu pro vývoj Business aplikací.

V praktické časti jsou na základě kritérií od odborníků z praxe a průzkumu vybrány ke zhodnoceni tři frameworků Angular, React a Vue. Následně jsou zhodnoceny dle kritérií vycházejících ze čtyř hlavních kritérií, které obsahuje podkritéria. Ke zhodnocení bylo využito průzkumu technologii ze webů State of JS 2021 a Developer Survey 2022 JS frameworků ke stanovení bylo na základě hodnocení odborníků z praxe.

Práce obsahuje ucelený přehled a problematiky vývoje business aplikací s využitím JS frameworků.

Klíčová slova: Single-page aplikace, Multi-page aplikace, JavaScript, Angular, React, Vue

Modern Javascript Frameworks

Abstract

The thesis deals with the evaluation and analysis of JS frameworks. The theoretical part deals with areas related to the practical part, namely the functionalities of JS frameworks and their superstructure for the development of Business applications.

In the practical part, three frameworks Angular, React and Vue are selected for evaluation based on criteria from practitioners and survey. They are then evaluated according to the criteria based on four main criteria which includes sub-criteria. The technology survey from the State of JS 2021 and Developer Survey 2022 JS frameworks were used to evaluate the frameworks based on practitioner evaluations.

This paper provides a comprehensive overview and issues of business application development using JS frameworks.

Keywords: Single-page application, Multi-page application, JavaScript, Angular, React, Vue

Obsah

1 Úvod

Od roku 1991, kdy veřejnost získala možnost používat WWW, prošel internet obrovskými změnami. Od první statické stránky, která obsahovala pár řádků textu, vytvořenou Timem Berners-Lee v době jeho zaměstnání v CERN-e, až po komplexní informační systémy, které v našich prohlížečích fungují dodnes. Tradiční fungování webových stránek a jejich prohlížečů spočívalo v získaní dát ze serveru, jejich zobrazení, případné úpravě, a poslaní zpět na server. Dnes se prohlížeče změnily na komplexní prostředí, které dokáže spouštět aplikace bez jakékoliv nutnosti instalace, bez ohledu na operační systém, na kterém běží. Tato schopnost umožnila migraci různých aplikací přímo z webu. Z velké části za to můžeme poděkovat popularitě JavaScriptu, ve kterém jsou tyto aplikace, často nazývané single-page aplikace, napsané. Single-page aplikace nabízejí řadu různých výhod, jedná se o komplexní aplikace dostupné pro uživatele kdekoliv, bez instalace, bez ohledu na zařízení a operační systém ze kterého se uživatel připojuje. Pro vývojáře to přineslo možnost vytvořit jednu aplikaci v jedné technologii, která je spustitelná kdekoliv, bez nutnosti úprav, případně úplného přepsání aplikace za pomoci technologie podporované jiným operačním systémem. Proto tento přístup může významně snížit náklady na vývoj, předávku a údržbu systému, a to jak náklady finanční, tak časové. Single-page aplikace se dají vyvíjet i za použití samotného JavaScriptu, tento proces je však poměrně náročný a zdlouhavý. Drtivá většina vývojářů sahá při nutnosti vytvořit single-page aplikaci po některém z frameworků, které nám nabízejí základní nástroje pro vytvoření takových to aplikací bez nutnosti je od začátku programovat. Tři nejpopulárnější JavaScriptové frameworky pro vytvoření single-page aplikací jsou v současné době Angular, React a Vue, a právě těmto frameworkům se budeme věnovat v této práci.[54][55]

2 Cíl práce a metodika

2.1 Cíl práce

Práce bude tematicky zaměřena na problematiku moderních JavaScriptových frameworků. Hlavním cílem diplomové práce je zanalyzovat a vyhodnotit vybrané JavaScriptové frameworky pro vývoji bussiness aplikací.

Dílčí cíle jsou:

- charakterizovat problematiku vývoje webových aplikací,
- analyzovat vybrané Javascriptové Frameworky,
- formulovat doporučení k nasazení JavaScriptových frameworků pro vývoj bussiness a webových aplikací.

2.2 Metodika

Teoretická část diplomové práce se bude zaměřovat na literární přehled odborných zdrojů, zaměřených na problematiku vývoje aplikací pomocí moderních JavaScriptových frameworků.

V praktické části budou vybrány konkrétní Javascriptové frameworky, které budou zhodnoceny dle zvolených kritérií. Na základě Vícekriteriální analýzy variant a Saatyho metodou budou získány výsledky. Dle výsledků budou formulovány doporučení pro jejich využití k vývoji bussiness a webových aplikací.

Na základě teoretické a praktické části dojde ke zformulování získaných poznatků a budou zpracovány závěry diplomové práce.

3 Teoretická východiska

Kapitola se zabývá teoretickými východisky a rešerší odborných zdrojů. Dané poznatky napomáhají k lepšímu porozumění problematiky vývoje webových aplikací, a to především v oblasti technologií souvisejících s jazykem JavaScript.

3.1 Charakteristika SPA

V případě použití SPA (single-page aplikací), je celá web stránka spuštěna jako jedna aplikace. V takovém případě se úplně oddělí prezentační a serverová vrstva aplikace, tzv. front-end a back-end, kde se SPA starají právě o front-endovou část. Při standartním designu web stránek, se při potřebě aktualizovat data zobrazená na stránce pošle na server požadavek, tzv. request. Server tento request zpracuje, vygeneruje výsledný HTML (Hyper Text Markup Language) kód, který odešle zpět prohlížeči. Prohlížeč tento kód vyrenderuje a zobrazí uživateli. Server se tedy stará i o prezentační vrstvu. V případě SPA se prezentační vrstva nachází v prohlížeči klienta. Při prvotním načítaní stránky se stáhnou nástroje potřebné na renderování stránek a vyrenderuje se první zobrazení, tzv. view. Jakmile je potřeba vygenerovat nové view, například pokud uživatel klikl na odkaz, který ho přesměroval na jiné místo v aplikaci, je tento view vyrenderován lokálně a dynamicky vložen do DOM stránky, zatímco původní, už nepotřebné, se z DOM (Document Object Model) odstraní. Spolu s vyrenderováním nového view se pošle request na server (jak je potřeba). Data se ze serveru většinou vrátí ve formátu XML (eXtensible Markup Language) nebo JSON (JavaScript Object Notation). Následně se zpracuje a doplní do view. Výsledkem je přesměrování a vyrenderování nové části stránky s aktualizovanými daty, bez nutnosti plného přenačítání v prohlížeči. [1]

Obrázek 1- SPA Lifecycle [42]

3.2 Technologie pro tvorbu SPA

Cílem SPA bylo vyhnout se nutnosti dodatečné instalace pluginů do prohlížeče musí tedy fungovat nativně v každém prohlížeči. Z tohoto důvodu na vytvoření SPA stačí základní technologie, které se využívají i na tvorbu standartních stránek – HTML, CSS (Cascading Style Sheets) a Javascript.

3.2.1 HTML

HTML je zkratka pro Hyper Text Markup Language. Tento jazyk je používaný na vytváření struktury webových stránek. Na vytvoření této struktury se používají HTML elementy, které jsou tvořené tagy. Tagy mohou být párové anebo nepárové. Jednotlivými elementy oddělujeme části obsahu, jako například hlavičku, odstavec, tabulky a podobně. Tyto elementy jsou poté prohlížečem vyrenderované do grafického zobrazení, které vidí užívatel. Momentálně nejnovější verzí HTML je verze 5.2. [2] [3]

3.2.2 CSS

Cascade Style Sheets, se používají na stylování HTML elementů. Fungují na principu selektorů, pomocí kterých vybíráme žádaný element, a ten poté stylujeme pomocí jednotlivých CSS vlastností, jako například font-size, pro velikost textu, nebo backgorund-color pro barvu pozadí elementu. Dnes nabírají na popularitě CSS preprocesory, jako například Sass anebo LESS, které přidávají do standartního CSS jazyka novou funkcionalitu, jako například používání proměnných, dědičnost a přehlednější zápis vnořených selektorů. Aktuální verze CSS je verze 3. [4] [5]

3.2.3 Javascript

Javascript je skriptovací programovací jazyk, používaný při tvorbě web stránek. Hlavní úlohou JavaScriptu je definovat a dynamicky upravovat chování stránky. Pomocí tohoto jazyka dokážeme manipulovat s HTML elementy, měnit jejich rozměry, pozice, obsah anebo CSS vlastnosti. Bez použití JavaScriptu není možné vytvořit jiné než obyčejné

statické webové stránky. Díky tomuto jazyku můžeme dnes vytvářet velmi složité aplikace, k nerozeznání od desktopových. Javascript začal jako čistě klientský jazyk, dnes už existují technologie, díky kterým můžeme použit Javascript k napsání kódu, který je spouštěn na serveru, příkladem je technologie *Node.js*. První verze jazyka vznikla v roce 1997, pod oficiálním názvem ECMAScript 1. Nejvýraznější změny přišly ve verzích ECMAScript 5 (zkráceně ES5) v roce 2009 a poté ve verzi ECMAScript 6 (ES6), které z JavaScriptu udělaly nejpopulárnější jazyk, který dnes známe. [6] [7]

3.2.4 Framework

„Frameworky JavaScriptu pomáhají překlenout rozdíly v prohlížečích a poskytují snadný přístup k jejich komplexním funkcím. Snaží se vyrovnat rozdíly v rozhraní a implementaci prohlížečů zabalením funkcí do nových aplikačních rozhraní. Obvykle nabízejí interakci s modelem DOM, podporu pro technologii AJAX (Asynchronous JavaScript and XML) a pomocné metody pro běžné úlohy.“ [8]

Aktuálně jsou dostupné stovky různých frameworků, které jsou běžně používané za nejrůznějšími účely. Je to instrument, který pomáhá vývojářům vytvářet dynamické stránky jednodušším způsobem. Vývojáři jím integrují do své aplikace různé předpřipravené funkcionality, jejichž vývoj by byl velice časově náročný, a v některých případech dokonce riskantní. Tento princip, kterému se říká deklarativní programování, využívají tzv. progresivní frameworky nebo knihovny. Ve velkých společnostech musí každý vývoj projít minimálně jedním testerem, což jej ovšem také prodražuje. Knihovny donucují developera udržovat určitou strukturu, což zjednodušuje provoz a odstranění případných chyb. Za jeden z nedostatků knihoven je považována špatná komunikace s vyhledávači, kdy např. Google ne ve všech případech dokáže webovou stránku najít. Tento problém je dnes více méně neaktuální, neboť vyhledávací systémy o něm věděly a s úspěchem tyto potíže nivelizují. [9]

Rozdíl mezí frameworky a knihovnami je velice subjektivní povahy. Frameworky poskytují komplexnější řešení, strukturu a koncepci kódů. Knihovny častěji pomáhají řešit nějaký konkrétní problém – například komunikaci se serverem apod. Jelikož *VUE.js*, *jQuery*, *React* i *Angular* mají obrovskou komunitu, tato řešení se postupem času velmi rozšiřují a je obtížné identifikovat, zda se jedná o framework, nebo knihovnu. Jak už bylo poznamenáno, *VUE.js* je *progresivní* knihovna neboli framework využívající principu

deklarativního programování. Tento princip používá ale také *React* nebo *Angular*. Z těchto důvodu v této práci budou používány pro pojmenování *Angular, Vue.js, jQuery* a *Reactu* společně pojmy knihovna a framework. [9][10]

3.2.5 jQuery

„*JQuery* je knihovna s otevřeným zdrojovým kódem, která poskytuje *funkcionální programovací rozhraní* k JavaScriptu. Jedná se o kompletní knihovnu, jejíž jádro je postaveno pomoci selektorů jazyka CSS pracujících s elementy modelu DOM.“ [8]

Používá se hlavně pro usnadnění komunikace s HTML dokumentem, ke zpracování události, vytvoření dynamického obsahu a vkládání grafických prvků do stránky (taby, stránkování…). Tuto knihovnu používají společnosti jako WordPress, Wikipedia, MaxCDN. Na principu *jQuery* jsou postavené také grafické frameworky, z nichž nejpopulárnější je *Bootstrap*.

Query() je základní funkcí této knihovny, která umí vyhledávat elementy na stránce, vytvářet nové elementy, manipulovat s nimi a mnoho dalšího. Vždy vrací objekt *jQuery*. Pro zjednodušení se používá zkrácený zápis této funkce – $(). Kupříkladu příkaz $('p'); vybere všechny paragrafy na sledované stránce. Předtím než začne provádět kód, musí být jisté, že všechny elementy obsažené v DOM se načetly, a to zajistí událost ready.

```
$(document).ready(function( { // tělo funkce } ) ) ;
```

Ukázka kódu 1 - Funkce

Podrobnější údaje jsou obsaženy v oficiální dokumentaci. [11]

jQuery je nejpoužívanější knihovnou na světě. Celkově ji využívá 66 179 046 webových stánek (v České republice 413 241). [49]

3.2.6 Node.js

Node.js je představitelem vysoce škálovatelného prostředí ve kterém lze spouštět Javascript a provádět *skripty* a to jak u klienta, tak na serveru. Byl vytvořen v roce 2009 Ryanem Dahlem. Jádro *Node.js* funguje na shodném principu jako v Google Chrome.

Aktuální verzi lze získat na oficiálních stránkách. Velikost stahovaného souboru je cca 30Mb. Součástí instalace je balíčkovací systém *npm*. Jedná se v podstatě o správce knihoven, pomocí kterého lze instalovat balíčky, řídit závislosti na projektech nebo sdílet vytvořený kód. V případě potřeby lze využít existující alternativy, mezi které se řadí např. Rhino, JSC, WSH a další.

Obrázek 2 - Node.js[43]

3.2.7 State Management

Označení *State management* lze doslovně přeložit jako správa stavu aplikace. Jedná se o proces, či metodu udržování znalostí o vstupech a stavu aplikace. Stavem jsou rozuměny veškeré informace ukládané danou aplikací, zpravidla s ohledem na předchozí interakce a události. Např. mezi vlastnosti DOM se řadí správa stavu formulářů bez nutnosti dalších interakcí (tzn. je schopen si zapamatovat vstupy od uživatele). S rostoucí složitostí aplikace a její komplexitou se může správa stavu stávat složitější a nepřehlednější. Toto platí obzvláště v situacích, kdy aplikace nabízí mnoho uživatelských akcí, které je potřeba nějakým způsobem spravovat. Proto by měl být kladen důraz na správný výběr nástrojů pro správu stavu již od počátku. Nástroje mohou určovat, kde dochází ke změně stavu, jaké s sebou změna přináší následky a tím i značně usnadnit budoucí řešení a analýzu chybových stavů. Nástroje pro správu stavu obvykle existují ve formě knihoven implementovaných do aplikace. V případě Angularu je správa stavu součástí již samotného frameworku. [50]

3.2.8 NPM a Node.js

Touto zkratkou je rozuměn Node Package Manager. Jedná se o nástroj, který zastupuje 2 věci. Za prvé se jedná o online úložné prostředí, skrze které jsou publikovány otevřené projekty *Node.js*. V druhé řadě jde o nástroj příkazového řádku, pomocí kterého lze s tímto úložištěm pracovat. Node.js package manager (NPM) je nápomocný při instalaci balíčku, řízení závislostí nebo při správě verzí. S pomocí manageru lze nainstalovat vybraný balíček zadáním jediného příkazu do příkazového řádku. Výhodou tohoto úložiště je, že je zde publikováno velké množství aplikací a knihoven a jejich počet stále roste. Publikované aplikace lze vyhledat na internetové adrese *http://npmjs.org/*. Vzhledem k tomu, že NPM je součástí *Node.js*, jsou po nainstalování prostředí *Node.js* oba tyto nástroje připravené k použití. [51]

Samotné *Node.js* prostředí je *multiplatformní* a otevřené. Umožňuje spustit javascriptový kód bez nutnosti spouštět ho ve webovém prohlížeči. Prostředí je postavené na JavaScript *enginu* V8 od Google Chrome. Prostředí slouží především k vývoji serverových částí webových aplikací. Mezi výhody *Node.js* patří jeho rychlost a schopnost obsluhovat velké množství připojených klientů zároveň. Díky těmto výhodám bývá často využíván pro tvorbu Aplication Programming Interface (API) serverů pro klientské SPA. Prostředí nabízí také velkou podporu knihoven různých JavaScriptových modulů, díky nimž je usnadněn vývoj webových aplikací. [52][53]

3.3 ANGULAR

Angular je framework používaný pro vytváření klientské části webových aplikací (tzn. *Front-end*). Charakteristickým znakem Angularu je, že na rozdíl od většiny ostatních frameworků, je sám napsaný v, a i při vytváření samotných aplikací se využívá, *Typescript*. *Typescript*, technologie od Microsoftu, je typovaný *superset* Javascriptu, který se následně kompiluje do čistého JavaScriptu.[12]

3.3.1 Typescript

Javascript je slabě typovaný jazyk. To znamená, že při jeho použití se nevykonává žádná kontrola použitého typu objektu nebo proměnné. Znamená to, že jsou povoleny

operace mezi proměnnými různých typů. Výsledkem tohoto typu programování je sice kompaktnější kód ,ale zároveň mohou nastat situace, které se velmi složitě odlaďují. Typickým příkladem může být například sčítání typu *string* a typu *integer*.

```
var x = "15";

var y = 15;

var z = x + y;  // "1515"
```

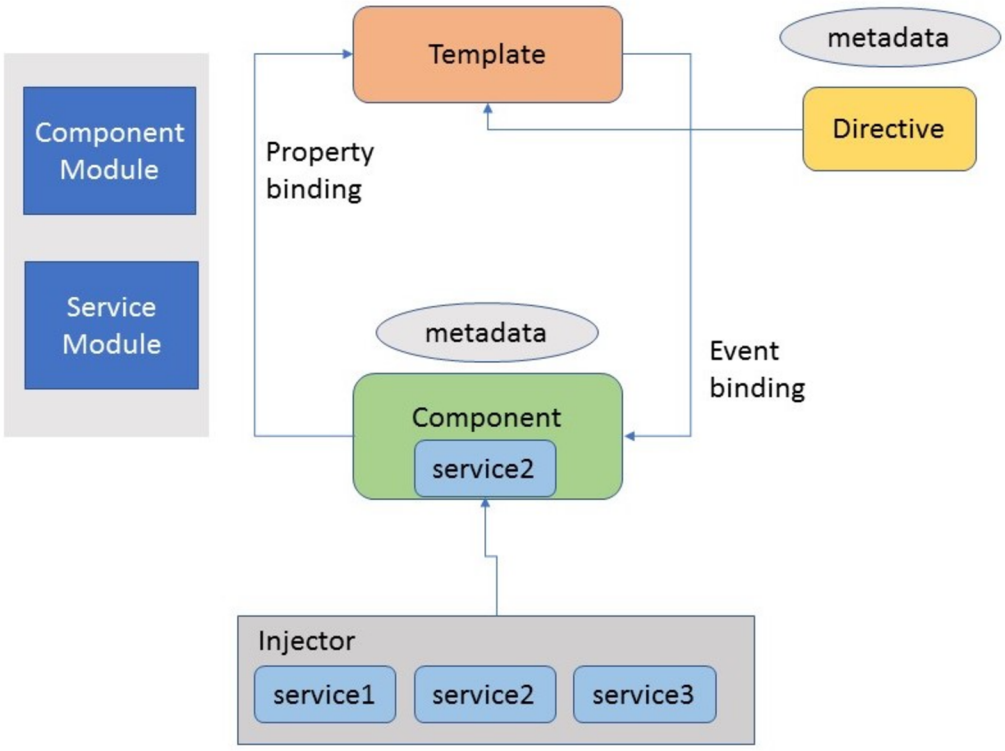
Ukázka kódu 2 - Typescript

V tomto případě bude výsledkem sčítání hodnota „1515“, definovaná jako *string*. Je to chyba, která se stává poměrně často a mnohdy trvá dlouho, než se ji pořadí odstranit. Právě zde přináší výhodu *Typescript*, který vnáší do JavaScriptu typovou kontrolu. Během kompilace TypeScriptu kompilátor kontroluje i jednotlivé typy proměnných, tím pádem by výše uvedený kód (napsaný v *Typescript*) neprošel kompilací. A díky různým pluginům, které se dají do vývojového prostředí doinstalovat, by byl programátor na nevalidní operaci upozorněn už při psaní kódu. Použití silného typování v TypeScriptu je zároveň dobrovolné. Variantou tak je ho nepoužít vůbec, nebo ho používat pouze v některých místech.[13]

3.3.2 Architektura Angularu

Základními stavebními kameny Angularu jsou tzv. *NgModuly*, které slouží jako kontext kompilace komponent. To znamená, že v modulech jsou deklarované jednotlivé komponenty

a rozdělují aplikaci do funkčních bloků. Komponenty představují jednotlivé views, ze kterých je skládána aplikace. Komponenty mají k dispozici služby, které se neváží na konkrétní *view* a mohou se používat v rámci celé aplikace. Tyto služby se nazývají service. Služby se používají v komponentách pomocí *Dependency Injection*. Pro navigaci v aplikaci se využívá *AnguleRouter*.[12][14]



Obrázek 3 - Architektura Angularu [44]

3.3.3 NgModuly a komponenty

NgModuly zabezpečují modulárnost každé *Angular* aplikace. Moduly jsou *kontejnery*, do kterých rozdělujeme související části kódu nebo jednotlivé části stránky, kde například každá položka v menu má vlastní *modul*. Každá aplikace obsahuje minimálně jeden kořenový *modul*, obvykle nazývaný *AppModule*. Tento modul se načítá při spuštění aplikace jako první.

V modulech se deklarují různé vlastnosti:

- *declaration*: deklarace jednotlivých komponent, *direktiv* nebo *pipe*,
- *exports*: deklarace, které jsou viditelné při importování modulu druhým modulem,
- *imports*: místo, kam přidáváme moduly, ze kterých chceme použít jeho exportované komponenty,
- *providers*: deklarace služeb,
- *bootstrap*: deklarace hlavní komponenty. Tento parametr by se měl v hlavním modulu použít pouze jednou. [15]

Komponenty tvoří jednotlivé views. Celé stránky se skládají z komponent, kde se každá komponenta stará o část zobrazení. Může se jednat o komponenty, které mají jen jednu úlohu, například se starají o zobrazení jedné položky listu nebo o komponenty, které slouží jako hlavní komponenty na stránce a jsou složené z většího množství jiných komponent.

Komponenta je *Typescript* třída, která je označena dekorátorem *@Component*. Komponenta se stará o logiku zobrazení a zpracování dat. Každá komponenta k sobě musí mít přiřazený HTML soubor, který slouží jako šablona. Tento HTML soubor může mít každá komponenta svůj vlastní, případně jí může být přiřazený už existující soubor jiné komponenty. Ke komponentě může být přiřazený i CSS soubor, který obsahuje styly k šabloně. Komponenta spolu se šablonou tvoří *view*. [16] [17]

Angular komponenty mají po svém vytvoření životní cyklus, tzn *Lifecycle hooks*. Jednotlivé metody se volají v určitých chvílích života dané komponenty. Jednotlivé metody se volají následovně:

- *ngOnChanges()*: volá se jednou před *ngOnInit()*, a vždy potom když se změní některá z dat, které jsou poslaná do komponenty z nadřazené komponenty pomocí *@Input(),*
- *ngOnInit()*: zavolané jednou po inicializaci komponenty potom, co se poprvé načítají data poslaná do komponenty pomocí *@Input()*.
- *ngDoCheck():* volané vždy poté, co *Angular* spustí svou detekci změn, vždy po *ngOnChanges()* a *ngOnInit()*.
- *ngAfterContentInit()*: spouštěné jednou, potom, co jsou načtená všechna externí data do šablony komponenty.
- *ngAfterContentChecked():* spouštěné vždy po spuštěné kontrole, která nastala po změně dat vázaných na komponentu.
- *ngAfterViewInit():* spouštěné jednou poté, co se zinicializuje šablona dokumentu.
- *ngAfterViewChecked():* spouštěné vždy po spuštěné kontrole a změně v šabloně komponenty.
- *ngOnDestroy():* spouštěné jednou předtím, než *Angular* odstraní referenci na danou komponentu. [16] [18]

3.3.4 Service, Dependency Injection

Service, neboli služba, je objekt nabízející společnou funkcionalitu, který podporuje výstavbu dalších bloků aplikaci například komponent nebo direktiv. Zjednodušují strukturu aplikace, zabraňují zbytečnému duplikování kódu, zlepšují opětovnou použitelnost kódu a zlehčují následné úpravy, protože stačí, když se provedou na jednom místě.

Příkladem může být service, která se stará o zjištění aktuálního kurzu mezi CZK a EUR. Vzhledem k tomu, že tato funkcionalita může být zapotřebí ve více komponentách, je vhodné vytvořit service a naimplementovat tuto funkcionalitu do každé komponenty zvlášť. V *service* vytvoříme metodu, která bude jako parametr brát do jaké měny chceme z CZK směnit. Druhý parametr bude suma, kterou potřebujeme směnit. Tato metoda pošle dotaz na server, který dokáže vrátit aktuální kurzy, tuto hodnotu zpracuje, a vrátí správný výsledek. Díky tomu, že je tato funkcionalita řešená jako *service*, je použitelná v rámci celé aplikace a není nutné ji definovat stále znovu. Služby se označují dekorátorem *@Injectable*.

V komponentách dokážeme *service* použít díky technice zvané *dependency injection*. *Angular* má tuto funkčnost zabudovanou jako svou základní součást. Pomocí této techniky vyjadřujeme závislost komponenty na *service*, kterou jsme vložili do konstruktoru dané komponenty. *Angular* při vytváření každé komponenty zkontroluje jeho konstruktor a vytvoří

objekty, na kterých je *konstruktor* závislý. V našem případě je tento objekt právě *service*. Tento objekt se ale vytvoří pouze v případě, že ještě neexistuje. To znamená, že *service* je tzv. *singleton*. Pro celou aplikaci existuje vždy jen jedna *instance* dané *service*. Tato vlastnost je velmi užitečná například když chceme sdílet data napříč aplikací mezi komponentami. [16] [18]

3.3.5 Routing

Většina aplikací potřebuje zobrazovat uživateli různý typ obsahu. Tento případ se řeší pomocí routingu, kdy se na základě URL adresy v prohlížeči mění zobrazovaný obsah. O obsluhu cest se stará modul *RouterModule*, proto každý *modul*, který pracuje s cestami musí importovat právě *RouterModule*. V Angularu se registrují jednotlivé cesty v příslušných modulech. Při vytvoření nového projektu je vytvořený kromě *AppModule*

v souboru app.module.ts i soubor app.routing.ts. V tomto souboru jsou definované jednotlivé cesty k modulům. Řekněme, že naše aplikace obsahuje cestu */auth*. V *app.routing.ts* vydefinujeme, že tato cesta odkazuje na modul *AuthModule*. Samotná obsluha cesty se už řeší v modulu *AuthModule*. V něm lze zadefinvoat, která komponenta se má spustit na dané adrese. Pokud chceme na adrese */auth/lo*fin spustit *LoginComponent*, vytvoříme konstantu ve tvaru

```
const routes: Routes = [
  { path: 'first-component', component: FirstComponent },
  { path: 'second-component', component: SecondComponent },
];
```

Obrázek 4 - AppRoutingModule [45]

Tuto konstantu potom importujeme do *AuthModule* a nová cesta bude přístupná v prohlížeči. [16] [20]

3.3.6 Data-binding

Data – binding je mechanismus, který se stará o koordinaci zobrazených dat a dat, nacházejících se v aplikaci. Existuje možnost přímo měnit HTML kód ale lepším, a přehlednějším řešením, je použít některou z vazeb, které nabízí *Angular*. *Angular* podporuje tři typy vazby dat:

- Jednosměrná ze zdroje do cíle: Data se zašlou do view nebo podřízené komponenty. Data v aplikaci nereagují na změnu, která přišla z cíle této vazby. Například vypsání proměnné do šablony.
- Jednosměrná z cíle do zdroje: Používá se například při spuštění metody po události, která nastala v šabloně – kliknutí na tlačítko, *interakce* se vstupem apod.
- Obousměrná: Data reagují na změny, bez ohledu na to, kde změnu nastala [21]

3.3.7 Šablony, pipe

Šablony tvoří viditelnou část aplikace. V architektuře MVC (Model-View-Controller), kterou *Angluar* používá, jsou šablony částí *view*, zatímco komponenta tvoří část *Model* a *Controller*. V *Angular* šablonách se používá klasický HTML jazyk obohacený o různé direktivy, které umožňují pokročilejší práci v rámci HTML. Důležitou funkcí *Angular* šablon je *interpolace*. S její pomocí dokážeme vypisovat hodnoty proměnných do HTML kódu. Používá se obalením názvu proměnné do dvojitých složených závorek.

```
<div>{{ variable }}</div>
```

Ukázka kódu 3 - Šablony

Uvedený kód vypíše hodnotu proměnné *variable* jako text do elementu . V rámci *interpolace* jsou validní i matematické výrazy. Další velmi používanou funkcí šablon jsou strukturální direktivy. Ty přinášejí funkčnost zobrazovat obsah na základě podmínek, případně vypisovat elementy z polí. Tyto direktivy mimikují podmínky a cykly, které známe z programovacích jazyků a přinášejí je do HTML kódu. Pro použití podmíněného zobrazení se používá *direktiva* *ngIf nebo *ngSwitch. Do parametru direktivy *ngIf vložíme podmínku a daný element se zobrazí pouze v případě, že je podmínka splněna.

```
<div *ngIf="condition">{{ variable }}</div>
```

Ukázka kódu 4 - ngIf

U tohoto zápisu se zobrazí daný element pouze pokud bude podmínka *condition* vyhodnocena jako *true*. V případě potřeby vypsat list se používá *direktiva* *ngFor. Tato *direktiva* je obrazem funkce cyklu *foreach*. Pro použití této direktivy zadefinujeme, jak má vypadat jeden prvek z listu. Šablona potom použije tento element a vypíše ho tolikrát, kolik je v listu položek.

```
<div *ngFor="let item of array">{{ item }}</div>
```

Ukázka kódu 5 - ngFor

Tento zápis říká – vyrenderuj *div element* tolikrát, kolik prvků je v listu *array*. Při každém jednotlivém výpisu ulož právě vypisovaný prvek do lokální proměnné *item*. Pomocí této proměnné dokážeme přistoupit k jednotlivým položkám listu a vypsat je. [16]

Součástí syntaxe používané v šablonách jsou také *pipe*. Pipe se využívají pro dynamickou transformaci dat do požadovaného formátu. Typickým příkladem je změna zápisu data z formátu JavaScript Date objectu, do formátu data tak, jak ho známe. Pokud vytvoříme nový Date object pomocí new Date() a pokusíme se ho vypsat do šablony pomocí *interpolace*, výsledná hodnota bude ve formátu:

```
Thu May 02 2019 13:26:58 GMT+0200 (Central European Summer Time)
```

Ukázka kódu 6 - Formát Datumu

Stačí ale přidat *pipe* s názvem date, kterou automaticky nabízí *Angular*. Pro přidání *pipe* stačí přidat její název za znak |.

```
<div>{{ variable | date }}</div>
```

Ukázka kódu 7 - Pipe

V tomto případě bude výsledná hodnota následující:

```
2.5.2019
```

Ukázka kódu 8 – Datum

Pipe dokáží přijímat i parametry. Pokud chceme zmíněné datum vypsat i s časem, stačí přidat parametr „*medium*“. [17] Zápis v tomto případě bude vypadat následovně:

```
<div>{{ variable | date:'long' }}</div>
```

Ukázka kódu 9 - Pipe2

Přímo *Angular* nabízí větší množství *pipe*, například pro formátování čísel, měn, písma a podobně. Další se dají doinstalovat ve formě balíčků, případně je možnost naprogramovat si vlastní. [24]

3.3.8 Vývoj frameworku

Angular je open source framework udržovaný společností Google. Jeho první verze byla vyvinutá slovenským programátorem Michalem Hevérym v roce 2009, v době, kdy pracoval ve firmě Brat Tech LLC.

Ve své první verzi se framework nazýval *AngularJS* a na jeho vývoj se využíval Javasript a ne *Typescript*. Začínal jako placená služba. Po počátečním neúspěchu se však tým v čele s Havérym rozhodl vydat *AngularJS* jako open source knihovnu. [25]

Angular od verze 2, která byla poprvé oznámena v roce 2014, prošel úplným přepsáním. Začal se používat *Typescript*, byl kladen velký důraz na modularitu a komponenty představují základní charakteristiku architektury frameworku. Vzhledem k velkým změnám nebyla tato verze zpětně kompatibilní s verzí *AngularJS*. Tento krok vzbudil velkou vlnu nevole mezi vývojáři. Finální verze 2.0 oficiálně vyšla v roce 2016. [26] [27]

Framework tuto počáteční nepřízeň ustál, aktuálně se nachází ve verzi 7.2 a těší se velké oblibě. Na portálu github.com, kde je hostovaný, má více než 47 600 hvězdiček. [28][66]

3.4 REACT

React je JavaScriptová knihovna, která se soustředí na *View* v klasickém MVC modelu. Jeden *view* je poskládaný jako hierarchie komponent. *React* sám o sobě není

plnohodnotný framework. Plnohodnotným frameworkem se stává až s přidáním dalších součástí, které z něho vytvoří kompletní technologii pro tvorbu SPA. Název *React* je odvozen od slova *reactive*, což naznačuje filozofii frameworku – jednotlivé komponenty reagují na aktuální stav aplikace. Aplikace jako celek neudržuje svůj stav, stavy si udržují pouze jednotlivé komponenty.

Pokud bychom chtěli přidat funkcionalitu na udržování globálního stavu, je k tomu potřeba externí plugin. React také v základu nenabízí něco jako *router*, případně obsluhu http requestů. O tom, co bude *React* podporovat rozhodne uživatel tím, které balíčky nainstaluje. Tento princip umožňuje uchovat celkovou velikost aplikace na nižší úrovni, jelikož obsahuje jen takové funkcionality, které programátor skutečně potřebuje a využije. [29] [30]

3.4.1 Komponenty

Komponenty nabízejí způsob rozdělení UI do nezávislých, znovu použitelných kódů. Koncepčně je komponenta v podstatě JavaScriptová funkce, která přijímá vstupní parametry jako *props* (zkratka slova properties) a vrací *React* Element popisující, co se má v prohlížeči vyrenderovat. [30] [31]

Komponenty mohou být v Reactu nadefinované jako třídy nebo funkce. Větší funkcionalitu nabídnou komponenty definované jako třídy. Pro nadefinování třídy jako *React* komponenty musí tato třída rozšiřovat třídu React.Component. [32]

Každá komponenta si udržuje svůj vlastní lokální stav, který nám umožňuje ukládat, upravovat nebo mazat vlastnosti uložené v komponentě. Ukázková komponenta v Reactu by mohla vypadat následovně:

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Obrázek 5 - React.Component [46]

Vytvořená třída Example rozšiřuje třídu React.Component. Do konstruktoru komponenty je zaslaný parametr *props*, čímž dojde k inicializaci vlastností komponenty.

Do objektu this.state přiřadíme počáteční stav komponenty. Zde konkrétně do vlastnosti date přiřadíme aktuální datum. Následně v metodě *render*() nadefinujeme, jak chceme, aby uživatel tento stav viděl ve svém prohlížeči. Podobně jako v případě *Angular* komponent, mají i *React* komponenty svůj životní cyklus. [29] [31]

- *render*(): jediná povinná metoda v rámci *React* komponenty. Volá se vždy při aktualizaci komponenty.
- *componentDidMount*(): volá se jednou poté, co byl vyrenderovaný výstup z komponenty.
- *componentWillUnmount*(): zavolá se jednou předtím tím, než je komponenta odstraněna z DOM. [29] [33]

3.4.2 JSX

JSX, celým názvem JavaScript Extension je doporučená technologie pro popis designu v Reactu. Jedná se o rozšíření syntaxe JavaScriptu. *React* nemá filozofii zpracování technologií do vlastních souborů, jako je to například v Anglularu, kde je *Typescript* a šablona psaná v HTML ve dvou samostatných souborech. Veškerý kód včetně JSX se v Reactu píše přímo do komponent. Na první pohled nápadně připomíná HTML jazyk. Ve výsledku je však JSX kompilován do JavaScriptového kódu. Jako příklad můžeme uvést výpis nadpisu do *render*() metody v komponentě.

```
<h1 className='large'>Hello World</h1>
```

Ukázka kódu 10 – ClassName

Tento kód bude ve výsledku přeložen do JavaScriptu jako:

```
React.createElement(
'h1',
{className: 'large'},
'Hello World'
)
```

Ukázka kódu 11 - createElement

Používání JSX není nutné. Je možné psát přímo JavaSriptový kód. Používání JSX ale výrazně zvyšuje přehlednost kódu a snižuje učící se křivku, jelikož je pravděpodobné, že většina webových programátorů se s HTML, kterému se JSX velmi podobná, někdy v minulosti již střetla. [34] [35] [66]

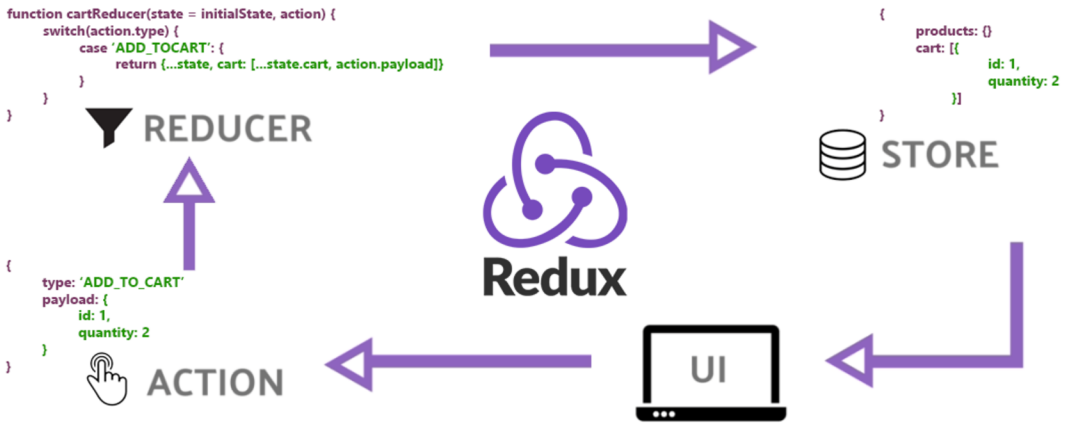
3.4.3 Redux

Redux je open source JavaScriptová knihovna, která umožňuje globální řízení stavu aplikace. *Redux* přináší jednosměrný tok dat a drží se několika zásad:

- aplikace má globální stav,
- změna tohoto stavu spustí aktualizaci dotčených komponent,
- měnit stav dokáží jen některé funkce,
- tyto funkce jsou spustitelné pouze uživatelem,
- vždy probíhá pouze jedna změna stavu.

Tento globální stav nedokáže sám od sebe spouštět žádné další akce. To umožní pouze vstup od uživatele. Díky tomu je tento stav mnohem snáze udržovatelný a zároveň je zabráněno jeho nechtěným změnám. [36]

V tomto globálním stavu mohou být uložené například odpovědi ze serveru, data v *cache* nebo lokálně vytvořená data. *Redux* umožňuje manipulaci a udržování právě tohoto typu dat.



Obrázek 6 - Getting started with Redux [47]

3.4.4 DOM

Za pomocí Javascriptu, pokud se jedná o manipulaci s DOM nelze vždy brát za jednoduchou a přímočarou. Časté problémy s výkonem vždy nemusí na žačátku ukázat rozdíl, ale na základě rozšíření aplikace již k problémům může dojít. Na základě těchto problémů byly vymyšleny tyto koncepty s manipulací DOM. A to Virtual DOM a Shadow DOM. Koncepty se tvořily na základě tvorby webových aplikací. Nástroje byly implementovány frameworky pro dané aplikace jako jsou *Angular Vue a React*.

3.4.5 Stínový DOM

Stínový DOM je funkce, která umožňuje webovému prohlížeči vykreslovat prvky DOM, aniž by je vložil do hlavního stromu DOM dokumentu. Tím se vytváří bariéra mezi tím, kam může vývojář a prohlížeč dosáhnout. Vývojář nemůže přistupovat ke stínovému DOM stejným způsobem jako k vnořeným prvkům, zatímco prohlížeč může vykreslovat a upravovat tento kód stejným způsobem jako u vnořených prvků. Dopad CSS rozprostřeného ve stínovém DOM určitého prvku spočívá v tom, že prvky HTML lze zapouzdřit bez rizika, že styly CSS uniknou a ovlivní prvky, které by ovlivnit neměly.

Přestože jsou tyto prvky zapouzdřeny s ohledem na HTML a CSS, mohou stále vyvolávat události, které mohou být zachyceny jinými prvky v dokumentu. Rozšířený podstrom v elementu se nazývá stínový strom. Prvek, ke kterému je stínový strom připojen, se nazývá stínový hostitel.

To usnadňuje také práci s CSS styly. Je možné používat jednoduché selektory pro konkrétní Komponenty a není potřeba psát globální CSS styly. Pokud potřebujeme určitou část překreslit, překreslí se jen část, která se změnila a není potřeba překreslení celého DOMu. To znamená, že se nezatěžuje tolik výkon aplikace. Hlavní výhodou je vývoj s pomocí Komponent, které na sebe navazují a předchází se zhoršení výkonu. [65] [66]

3.4.6 Virtual DOM

V praxi znamená použití Virtual DOM nacachované verze DOM aplikace, která je uložená v paměti zařízení, na kterém běží. O aktuálnost zobrazení v paměti s reálně vyrenderovaným zobrazením se stará knihovna ReactDOM. Výhodou tohoto principu

je to, že se vždy aktualizují jen ty části stránky/komponenty, u kterých je to nutné, tzn. došlo u nich ke změně stavu. Tím odpadá nutnost znovu vykreslovat celou stránku. [62]

3.4.7 Rozdíly stínového a virtuálního DOM

Tyto funkcionality se zaměřují přímo na problémy výkonu, kde si svojí vlastní cestou tvoří dodatečnou instanci DOM (Objektového modelu dokumentu) Virtuální DOM je kopie DOMU, kde probíhá porovnávání DOMů mezi sebou při aktuální změně. DOM stínový je vytvářen po meších částech a je oddělen od zbytku částí. Tato funkcionalita je výhodou pro práci s komponentami. [62]

3.4.8 Aplikování stylů do komponent

React nabízí velkou volnost při výběru, jak budeme stylovat jednotlivé komponenty. První možností je vytvoření souboru, který obsahuje kód na stylování a jeho následné naimportování do komponenty. Tento soubor může být typu CSS, ve kterém využijeme klasický CSS kód nebo lze využít některý z preprocesorů, jako jsou například SASS či LESS.

Další možností je vytvoření stylů ve formě JavaScript objektu a následné přiřazování stylů z tohoto objektu přímo do šablony JSX. Toto řešení přináší výhody, například ve formě skutečnosti, že máme kontrolu nad tím, kde se které styly využívají pomocí počtu referencí na objekt, ve kterém se styly nacházejí nebo jednoduché změny stylů na základě změny hodnoty proměnných v komponentě. Při tomto využití ale přicházíme i o výhody, které přináší preprocesory, například využití CSS proměnných nebo skládání jednotlivých selektorů. [37]

3.4.9 Vývoj frameworku

React je open source knihovna vytvořená a udržovaná společností Facebook. Hlavním motivátorem vzniku byla enormní expanze Facebooku, jak z pohledu počtu uživatelů, tak narůstající složitostí funkcionalit této sociální sítě. Z těchto důvodů začalo být udržování aplikace postupem času složitější.

React byl představen jako open source projekt na konferenci JS ConfUS v roce 2013. Z počátku panovaly o této nové technologii pochybnosti, především proto, že šla proti všem do té doby, zažitým postupům. [37]

React neprošel tak velkými změnami jako například *Angular* a drží se své původní filozofie. Po původních pochybnostech se v dnešní době řadí mezi nejpoužívanější a nejpopulárnější JavaScriptový SPA framework, který má na github více než 128 000 hvězdiček.

3.5 VUE

Vue je JavaScriptová knihovna pro tvorbu SPA. Ze třech zmíněných frameworků je nejmenší. Drží se zhruba návrhového vzoru MVVM – Model – View – ViewModel. Tento vzor určuje rozdělené *View* (uživatelského rozhraní a Modelu (dat, se kterými uživatelské prostředí pracuje). Komunikaci mezi těmito vrstvami *VievModel*, v tomto případě zajišťuje knihovna *Vue*. Na rozdíl od *Angularu*, je možné *React* i *Vue* nasadit do již existujícího projektu. Díky relativní jednoduchosti základní verze knihovny je možné vytvořit jednoduché komponenty, které obsluhují jen část už existující *web* stránky. Je však možné rovněž vytvořit komplexní SPA díky doplnění základní knihovny oficiálními pluginy, které z knihovny vytvoří plnohodnotný framework. Tyto pluginy se starají o *routing*, udržování globálního stavu apod. V tomto směru si je *Vue* velmi podobné s *Reactem*, využívá však také některé schopnosti *Angularu* (např. obousměrnou vazbu dat). [39]

3.5.1 Komponenty

Komponenty jsou znovu použitelné *instance* knihovny *Vue*. Každá komponenta může obsahovat několik metod nebo nastavení. Jedná se například o jednotlivé metody životního cyklu, funkce pro vytvoření vlastních metod v rámci komponenty nebo metod pro uložení dat.

```
export default {
  name: 'app',
  components: {
    Sidebar
  },
  data: () => ({
    showNavigation: false,
    showSidepanel: false
  })
}
```

Ukázka kódu 12 - Vue – Komponenty

Na příkladu uvedeném nahoře můžeme vidět příklad *Vue* komponenty. Tato komponenta je v aplikaci dostupná pod jménem app. V objektu components se registrují podřízené komponenty, které jsou zobrazeny v rámci dané komponenty. Funkce data () obsahuje deklarace proměnných, se kterými se v komponentě pracuje. Data se musí uchovávat v rámci funkce z důvodu uchování jedinečné *instance* těchto dat pro každou instanci komponenty. V případě, že bychom měli tu samou komponentou použitou vícekrát a data by byla uložena jako jednoduchý objekt, ovlivnila by změna dat v jedné komponentě i data v ostatních instancích. [40]

Podobně jako v *Reactu* můžeme do komponenty poslat z nadřazené komponenty data pomocí props. V *Angularu* docílíme podobného sdílení dat pomocí funkce *@Input ()*. Ve *Vue* komponentách se píše JavaScriptový kód, kód HTML i se využívá stylování pomocí CSS nebo libovolného preprocesoru, jako jsou například SASS nebo LESS.

Jak již bylo řečeno, komponenty mají rovněž svůj životní cyklus. Tento cyklus se skládá z několika metod:

- *beforeCreate()*: Proběhne jednou, před inicializací komponenty.
- *created():* Proběhne jednou po inicializaci dat z objektu data.
- *beforeMount():* Proběhne jednou předtím, než dojde k prvnímu vyrenderování šablony.
- *mounted():* Proběhne jednou po kompletním vyrenderování komponenty.

- *beforeUpdate():* Proběhne vždy po změně dat v komponentě, předtím než dojde k opětovnému vyrenderování šablony komponenty.
- *updated():* Proběhne po vyrenderování po změně dat v komponentě.
- *beforeDestroy():* Proběhne před odstraněním komponenty z DOM pokud jsou data a šablona komponenty ještě plně dostupné.
- *destroyed():* Proběhne jednou potom, co byla komponenta odstraněna. [41]

3.5.2 Šablony

Vue poskytuje svobodu v tom, jakým způsobem se píše uživatelské rozhraní. Je možnost použít technologii JSX, podobně jako je to v případě technologie *React*. Je však možné použít i tradiční jazyk HTML spolu s direktivami a dalšími funkcemi, které framework nabízí. *Vue* direktivy poskytují bohatou funkcionalitu přímo v rámci HTML jazyka, podobně, jako je tomu v případě frameworku *Angular*. Tyto direktivy umožňují ovládat *interakci* s uživatelem, vypisovat hodnoty proměnných pomocí *interpolace*, vypisovat obsah listů pomocí ekvivalentu funkce *foreach* apod. Direktivy začínají označením v-, za kterým následuje název direktivy a případný parametr. Jako příklad může posloužit direktiva obsluhující klik uživatele na nějaký element na stránce:

```
<button v-on:click="counter += 1">Add 1</button>
```

Ukázka kódu 13 - Šablony

Na elementu tlačítka byla použita direktiva v-on s parametrem click. Ta zaručuje, že po kliknutí na tlačítko se vykoná kód v uvozovkách. Může obsahovat zavolání funkce nebo, jako je tomu v tomto případě, přičte jedničku k proměnné counter. Pro vypsání listu a simulace funkce *foreach* se dá použít direktivy v-for.

```
<ul>
  <li v-for="item in items">
    {{item.message}}
  </li>
</ul>
```

Ukázka kódu 14 - v-for

Pomocí této direktivy projdeme všechny prvky listu items a pro každou iteraci vypíšeme položku message.

3.5.3 Sledování změn

Vue nabízí funkci sledování změn dat v komponentě. Tuto funkci umožňují tzv. *watcher*, které definujeme v objektu *watch* v jednotlivých komponentech.

```
watch: {
variable: function (newVar, oldVar) {
console.log(newVar, oldVar)
}
}
```

Ukázka kódu 15 - watch

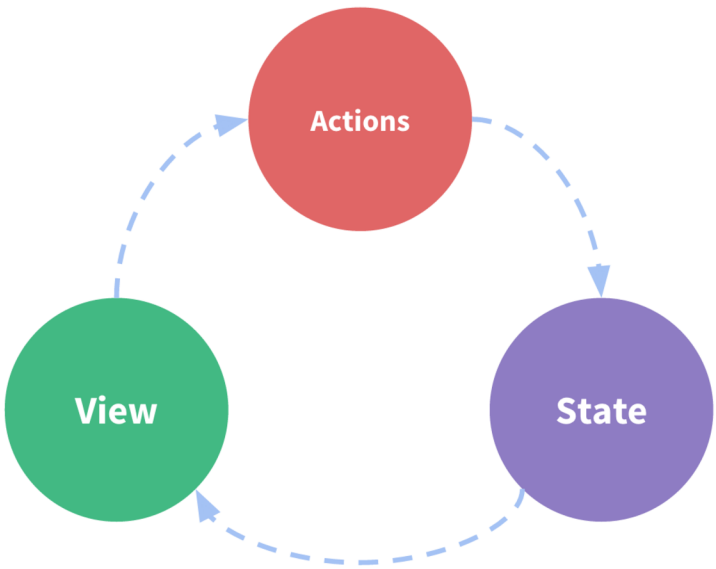
Na příkladu kódu výše nadefinujeme funkci, která se spustí vždy po změně proměnné *variable*. Jako parametry této funkce jsou dvě proměnné. V prvním parametru je uložena nová hodnota po změně, v druhém parametru je pak původní hodnota. Tato funkcionalita je vhodná například tehdy, když potřebujeme vykonat akci po změně dat uživatelem, například jí požadujeme odeslat na server a uložit. [39]

3.5.4 Vuex

Vuex je oficiální knihovna sloužící pro udržení globálního stavu. Jde o podobnou aplikaci jako v případě využití *Reduxu* s *Reactem*. Jedná se o centrální úložný prostor pro všechny komponenty v aplikaci zabezpečující změnu stavu předvídatelným způsobem.

Používá se v případech, kdy na jednom stavu dat aplikace závisí více komponent nebo například pokud akce z různých částí aplikace potřebují změnit stejná data. Při použití této knihovny odpadává nutnost zdlouhavě posílat data do podřízených komponent pomocí *props*. Tento systém přenášení dat například vůbec nefunguje pro komponenty na stejné úrovni. Díky *Vuex* vyjmeme tato společná data nad komponenty a vytvoříme z nich jeden *singleton*, ke kterému mají komponenty přístup a mohou v něm data měnit pomocí mutací (data se nedají měnit přímo). Na rozdíl od *Reduxu* je však *Vuex* vytvořený speciálně pro framework *Vue*. Tato knihovna funguje na základě několika konceptů:

- *State*: objekt, který obsahuje všechna data, která mají být dostupná v rámci aplikace.
- *Getter*: funkce používaná pro přístup k datům uloženým v *State*.
- *Mutations*: slouží k obsluze změny stavu v *State*.
- *Actions*: Funkce využívané k vyvolání mutací.
- *Modules*: umožňuje rozdělit jeden globální stav aplikace na větší množství menších, což u velkých aplikací umožní lépe udržovat přehlednost kódu.



Obrázek 7 - What is Vuex [48]

3.5.5 Vývoj frameworku

První verze frameworku začala být vyvíjena v roce 2013 programátorem Evanem You, který byl v té době zaměstnancem společnosti Google. Původní plán byl zachovat

všechny oblíbené schopnosti *Angularu*, odstranit koncepty, které vyžadují dlouhé studium potřebné pro jejích využívání a snížit jeho velikost a robustnost.

Verze 1.0 vyšla v roce 2015. Na rozdíl od *Angularu* a *Reactu* zaznamenala okamžitý úspěch. Krátce po vydání si jako svou oficiální frontendovou technologii vybral *Vue* také velmi známý a oblíbený framerwork *Laravel*.

Verze 2.0 byla vydána v roce 2016. Nejvýznamnější změnou bylo přidání renderování pomocí technologie Virtual DOM. [39]

V současné době se *Vue* nachází ve verzi 2.6 a na hostujícím portálu github.com má více než 138 tisíc hvězdiček.

3.6 Frameworky a knihovny

Frameworky a knihovny jsou stacky kódů, které ulehčují práci. Vznikají na základě zjednodušení práce s Javascriptem jako takovým. Využívání může mít spousty výhod ale i nevýhody. Můžeme říct, že u některých frameworků, které se často neaktualizují mohou zanechat problémy s velikostí konečné aplikace. Frameworky nebo knihovny vznikají k pře používání různých funkcionalit, aby si nemuseli řádky těchto kódů pořad psát znovu a znovu a tím pádem by aplikace nabývala na velikosti. Tyto knihovny často udržuje komunita, a to komunita vývojářů.

U frameworku je zásadní také architektura a knihovna architekturu nemá. Jak je známo tak framework se často skládá z nezávislých knihoven. Někdy vývojáři oznamují to, že knihovna je brána jako framework, a to na základě spojení více knihoven do jednoho stacku kódu. [56][57] [66]

3.7 Průzkum a studie

Tato kapitola informuje o výzkumech, dle kterých byly vybrány poznatky, díky kterým měli větší vliv na zpracování této práce. [56][57] [66]

3.7.1 Developer Survey 2022 Stack Overflow

V květnu roce 2022 probíhal průzkum stránek Stack Overflow zaměřující se na vývojáře, jak se učí a zvyšují úroveň, jaké nástroje používají a co chtějí. Zapojilo se téměř 70 tisíc respondentů. Výzkum probíhal během února 2022. [56][57] [58][59][60] [66]

První část průzkumu se zabývala zjištěním obecných informací o respondentech, které se týkaly především:

- Role v pracovním poměru,
- aktuální zkušenosti,
- výše vzdělaní.

Druhá část pak byla zaměřena na technologie. Otázky kladené respondentům byly [56][57] [58][59][60] [66]:

- nejoblíbenější známé technologie,
- žádané technologie,
- nejoblíbenější placené technologie.

Další důležitou a kladenou otázkou byla na průzkum trhu a její problematiky. [56][57] [58][59][60] [66]

Zkoumáno bylo:

- informace o zaměstnání,
- kariéra respondentů,
- poptávka práce,
- priority v zaměstnání,
- celková mzda.

Prvním výstupem této práce byl výsledek průzkumu. Jednalo se hlavně o to, co je pro vývojáře důležité v pracovním životě. Hlavním hlediskem, na kterém se vývojáři shodli bylo z 57,5 % to s jakým jazykem, případně frameworkem by měli pracovat. Důležitou věcí bylo také to, s jakou technologií vývojáři pracovat nechtějí. To byl hlavní výstup pro Manažery společnosti zaměstnávající právě vývojáře. Důvodem může být hlavně nedostatečná

dokumentace anebo složitost frameworku, a proto často vývojáři nechtějí s takovýmto frameworkem nebo knihovnou pracovat. Proto vývojáři hledají alternativní náhrady těchto knihoven.

Finální průzkum byla samotná komunikace na Stack Overflow. Zkoumány byly hlediska [56][57] [58][59][60] [66]:

- frekvence používání webu Stack Overflow,
- účast určitého průzkumu.

3.7.2 State of JS 2022

Každý rok, posledních 6 let probíhá průzkum technologií založen Javascriptovým jazykem. V roce 2022 vyplňoval dotazník 17 138 jedinců. V rámci měření v roce 2022 bylo získáno mnoho odpovědí od vývojářů, celkově z 137 zemí. [61][62] [63] [66]

Obrázek 8- Demografie průzkumu State of JS 2022 [61]

Primární bylo získat názory a informace o Javascript technologiích. Informace se získávali pro testovací nástroje. Byli jimi různé IDE, knihovny, které je spojovali. [61][62] [63] [66]

Na výběr bylo z pěti možností:

- „Nepoužil(a) bych“
- „Nezajímám se“
- „Použil(a) bych“
- „Chci se naučit“
- „Neznám“

Odpovědi byly poté zpracovány a graficky zobrazeny. V následující tabulce byly shrnuty výsledky na rok 2022. [61][62] [63] [66]

	Spokojenost	Zájem	Užití	Známost
Alpine	79%	33%	6%	48%
Angular	45%	16%	54%	100%
Ember	21%	12%	9%	87%
Lit	77%	40%	7%	38%
Preact	74%	32%	14%	80%
React	84%	48%	80%	100%
Stimulus	62%	21%	2%	19%
Solid	90%	56%	3%	38%
Svelte	90%	68%	20%	94%
Vue	80%	50%	51%	100%

Tabulka 1- Frontendové Frameworky [62]

Můžeme vidět, že *React*, *Angular* a *Vue* jako nejpopulárnější frameworky. Nejvíce se používá *React*. Dnešní době se skloňuje framework *Svetle*. Je o něj velký zájem. Patří k velice jednoduchým a novým způsobům vývoje pomocí Javascript jazyku. Jak je zobrazeno v tabulce výše nejméně oblíbený framework je *Angular* a to na základě jeho složitosti funkcionalit. [61][62] [63] [66]

3.8 Použité nástroje k měření výkonu

K porovnání výkon stejných aplikací psaných ve frameworcích (knihovnách) *Angular, React a Vue* slouží nástroj *js-framework-benchmark*[1].

Scénáře, které se testují jsou tabulkové operace. Záleží na práci s daty v řádcích. Jelikož jsou frameworky dosti podobné i výsledky se očekávají podobné, jelikož pracují s *DOM*, tedy HTML stromovou strukturou aplikace. Tento nástroj byl implementován vývojářem Stefanem Krausem pocházející z Německa. Tento nástroj má jasně daná pravidla, které se při testování musí dodržovat. Jak již bylo řečeno na základě testů lze tedy říct, že *Vue* oproti *Angularu* a *Reactu* má lepší práci s Alokací paměti, to znamená menší množství paměti. Pokud máme jednoduchou stránku, která se updatuje tak lepší výkon ukazuje framework *React*. Je rychlejší než *Vue* a *Angular*. Po těchto zjištěních lze říct, že tento nástroj je velice oblíbený a přesný. [64] [65] [66]

[1] *Dostupnost na https://github.com/krausest/js-framework-benchmark*

4 Vlastní práce

Diplomová práce ukazuje analýzu a výběr JS frameworků. Dle vybraných kritérií spadající pod určité aspekty se zhodnocovali tři vybrané frameworky.

4.1 Výběr frameworků

Výběr byl zvolen na základě odborníků z praxe a potřeb vývojářů. Jelikož každý rok najdeme nové frameworky, které se snaží získat nové vývojáře. Ke zhodnocení museli frameworky brát v potaz:

- vyspělost,
- open-source,
- používání v praxi,
- udržovatelné a aktivní.

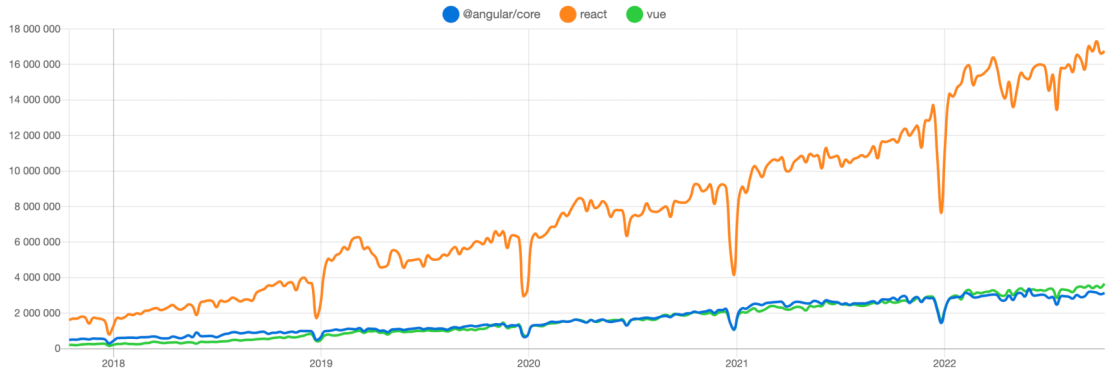
Kontribuce je brána jako označení MIT. Jedná se o projekty open-source. Hodnocení bylo závislé na datech průzkumu. Průzkum byl zaměřeny na vývojářské komunity a odborné zdroje. Finálně byly vybrány tři nejvíce používané a vyhovující frameworky ke zhodnocení. V dnešní době existuje velké množství frameworků. Výběr právě třech frameworků i s předem vydefinovanými kritérii byl velice obtížný. Hlavním materiálem byl využit dotazník State of JS 2021. Tento dotazník každý rok vyplňují vývojáři. V širším výběru figurovaly frameworky *Alpine, Angular, Ember, Lit, Preact, React, Stimulus, Svelte a Vue*.

	Verze	Web	GitHub	Licence
Angular	14.2.7	angular.io	angular/angular	MIT
React	18.2.0	reactjs.org	facebook/react	MIT
Vue	3.2.41	v3.vuejs.org	vuejs/vue-next	MIT

Tabulka 2 - Vybrané frameworky [54]

4.1.1 Kritéria výběru

Vybrané byly tři frameworky. Dalším krokem bylo zvolit kritéria ke zhodnocení. Ty se volila dle průzkumu State of JS 2021. Další nástroj pro výběr byl *npm trends*. Výstupem byla časová řada (viz Obrázek 9). Na obrázku je vidět počet stažení frameworků za posledních 5 let.



Obrázek 9 - Vývoj stažení frameworků [54]

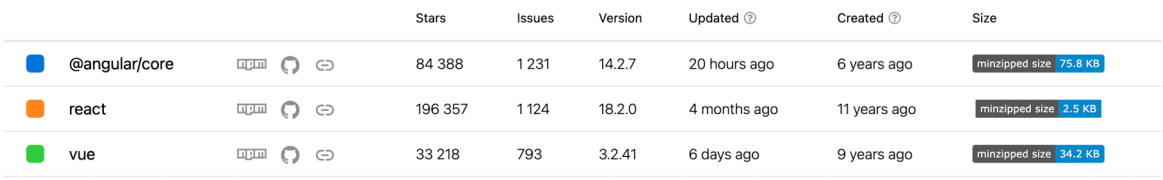
Na obrázku je vidět, že nejvíce využívaný a stahovaný framework je *React*. Pokud se podíváme detailněji na obrázek, tak vidíme velký pokles stahování každoročně na konci roku.

Na grafu můžeme vidět, že jedno seskupení tvoří *React*, další *Vue* a *Angular*. Výsledek je z průzkumu State of JS 2021.V (Tabulce 1) vidíme využívání vybraných frameworků. V tabulce (Tabulka 3), vidíme poslední data stažení z 20.10.2022.

Angular	React	Vue
3 194 427	17 560 702	3 782 547

Tabulka 3 - Počet stažení [54]

Obrázek (Obrázek 10) ukazuje výstupy z *npm trends*. Najdeme zde informace o udržovatelnosti, oblíbenosti, vyspělosti a použití.



	Stars	Issues	Version	Updated	Created	Size
@angular/core	84 388	1 231	14.2.7	20 hours ago	6 years ago	minzipped size 75.8 KB
react	196 357	1 124	18.2.0	4 months ago	11 years ago	minzipped size 2.5 KB
vue	33 218	793	3.2.41	6 days ago	9 years ago	minzipped size 34.2 KB

Obrázek 10 – Informace o frameworku [54]

Některými způsoby můžeme vyjádřit oblíbenost. Týdenní počet stažení, Google trendy nebo hodnocení na GitHubu můžeme považovat za nejčastější využívanou charakteristiku, a to na základě odborných zdrojů. (viz Tabulka 4).

Víme, že dnes vyspělé frameworky, které jsou nejvíce využívané, patří k oblíbeným. Využité byly míry spokojenosti a využívání (viz Tabulka 2). Z dostupných dat byla využita kritéria zohledňující vyspělost, použití a oblíbenost.

Mezi kritéria výběru byl zahrnut:

- rok vydání frameworku,
- hodnocení na GitHubu,
- týdenní počet stažení,
- míru spokojenosti,
- míru užití.

4.1.2 Vybrané frameworky

React

Framework byl nasazen na Facebook News Feed v roce 2011. V roce 2013 se *React* stal otevřeným zdrojem, a během JS ConfUS, který se konal od 29. do 31. května, představil Jordan Walke *React* světu. Byl ovlivněn XHP, knihovnou komponent HTML pro PHP. *React* v0.13 byl vydán v březnu 2015. Tato verze *Reactu* měla vítanou novou funkci, kterou byla podpora tříd ES6. Ve stejném měsíci vydání *React* v0.13 byl světu představen *React Native*. Knihovnu *React* jak víme využívá z velké části Facebook, ale i Instagram a WhatsApp. Knihovnu také nalezneme využitou v aplikacích, které velice dobře známe a to Twitter, PapPal, Uber a Netflix. Aktuální verze je 18.2.0

Vue

Vue každým rokem získával na popularitě. *Vue* framework, který byl implementován v roce 2014 bývalým zaměstnancem Googlu jménem Evan Youe, je z velké části tento framework podporován Patreonem. Jako jeho nevýhodu můžeme chápat to, že ho nepodporuje žádná významná společnost. Další známé společnosti, využívající *Vue* ve svých projektech, jsou WizzAir, Xiaomi, Alibaba. Nejaktuálnější je verze 3.2.41.

Angular

Aktuální verze je 3.2.41. V roce 2009 založila společnost Google tento framework který vystupuje pod licencí dostupnou jako MIT. Tento Framework již od roku 2022 není rozšiřován a udržován. Doporučuje se přejít na jinou verzi *Angularu*. Dnes máme framework rozdělen na *AngularJS* a *Angular*. Toto rozdělení vzniklo v roce 2016 a byla to velká změna pro všechny vývojáře. Aktuální verzi *Angularu* je 14.2.7. *Angular* dnes můžeme najít aplikacích Microsoft a Google.

4.2 Vybraná kritéria

Vybrány byly tři frameworky. Aby se mohli frameworky zhodnotit bylo nutné vybrat vhodná kritéria. Kritéria byly vybrané čtyři a to vývoj, použitelnost, výkon a komunita. Komunitou je myšleno její velikost. To znamená, kolik vývojářů se podílejí na kontribuci daného frameworku. V potaz se brali existující aplikace, které komunita spravovala.

Kritéria zhodnocení:

- Výkonu,
- Použitelnost,
- Vývoj,
- Komunita a její velikost.

4.2.1 Výkon

Pokud potřebujeme webovou aplikaci načíst, výkon je pro vývojáře velice důležité hledisko. V rámci běhu aplikace na stává že aplikace zpracovává no požadavků. Toto se projevuje při načítání mnoha dat z S logické části aplikace. Pokud pracujeme s jednu stránku aplikaci při prvním běhu je potřeba načíst všechna data najednou. Single page aplikace se zakládají na práci s (DOM) objektové model dokumentace. Pro nás výkon této práci znamená, jak nahlížet na framework ve kterém jsou aplikace napsány. Důležité je také pro aplikaci, pokud je správně zoptimalizovaná výkon bývá mnohem větší. Pokud máme aplikaci z mnoha nepotřebnými řádky, které jsou v aplikaci zbytečné, může aplikace být pomalejší. Proto se hodnotili aplikace, které byly psaný stejně a používali se stejně nástroje. Výkon byl hodnocen nástrojem js-framework-benchmark. (viz Příloha 1).

Kritéria:

- manipulace s DOM,
- spouštěcí metriky,
- alokace paměti.

4.2.2 Použitelnost

Použitelnosti je velice široký pojem. Toto hledisko můžeme brát jako přístup k dokumentaci a nečitelnost frameworku. Proto použitelnost bývá velice důležitá pro vývojáře. A to hlavně z pohledu jednoduchosti a na nečitelnosti. Pokud má Framework špatně vedenou dokumentaci je těžké se v něm na první pohled vyznat a pracovat s ním.

Průzkumem stránek Stack Overflow bylo zjištěno chování vývojářů při určitých problémech v posunutí se v práci dále. Z 93 % bylo zjištěno, že k zjišťováním odpovědí se používají stránky Stack Overflow (viz Příloha 2). V dnešní době bývají projekty open-source tzn., že vývojáři píši zpětnou vazbu na samotnou dokumentaci. Proto je brána použitelnosti za důležité hledisko pro vývojáře, aby daný framework byl co nejrychleji k naučení a používání.

Kritéria:

- dokumentace,
- křivka učení.

4.2.3 Vývoj

Vývoj chápeme více směry. Jedním je pohled práce vývojáře s frameworkem. Jestli je dobré ho využívat v aktuálním existujícím projektu. Nebylo jednoduché vybrat to hledisko. Důležité bylo pochopit tak hledisko vnímat a hodnotit ho.

Na různých internetových portálech bylo často odkazováno na ekosystém a flexibilitu. Pokud ekosystém frameworku je na velice dobré úrovni, zjednoduší vývoj aplikace a sníží náklady právě na vývoj.

Pokud framework má velice propracovaný ekosystém vývoj bývá často velice rychlý. Flexibilita nám říká jinak lze framework používat volněji a jestli je možné ho používat již existujícím projektu. Hlavním důvodem je, jestli vývojář musí držet pravidla frameworku nebo má volnou ruku k vývoji.

Kritéria:

- dostupné knihovny,
- ekosystém,
- flexibilita.

4.2.4 Komunita

Komunita je velice důležité hledisko. Pokud vývojáři přemýšlí nad implementací, často naleznout odpověď uvnitř komunity využívaného frameworku. Odpovědi na otázky dané komunity nalezneme na portálu Stack Overflow. Bylo zjištěno, že Stack Overflow je využíván denně s 28,6 % respondentů na denní bázi a 30,5 % skoro každým dnem. (viz Příloha 3).

Dnes již můžeme na internetových portálech vyhledat spoustu výukových materiálů, tzn., že se dokumentace využívá méně. Jedním z hlavních portálů je kanál YouTube,

kde lze dnes nalézt spousty kurzů. Dalším známým portálem je UDEMY, kde nalezneme placené tutoriály. Na základě velikosti komunity bylo vybráno toto hledisko ke zhodnocení vybraných kritérií. A to hlavně díky její velikosti. Hlavní vlastnost komunity je aktivita a rozsah. Velikost komunity zjistíme na základě počtu vývojářů, který se o komunitu frameworku zajímají a na kterých webových stránkách ji můžeme najít. Někdy lze ani velikost komunity nestačí. Velikost komunity může chápat každý různými způsoby. Pro někoho to může být 100 lidí pro někoho 5lidí. Velikost komunity jsou k nalezení na webu jako je *Reddit* a Stack Overflow.

Kritéria:

- velikost komunity,
- aktivita komunity.

4.2.5 Vybrané kritéria

Na základě internetových článku a názor komunit bylo nakonec zvoleno devět kritérií. Každé z kritéria vždy řadíme k jednomu ze čtyř vybraných hledisek. Tabulka (viz Tabulka 4).

Hledisko	Kritérium
Výkon	Manipulace s DOM
	Spouštěcí metriky
	Alokace paměti
Použitelnost	Dokumentace
	Křivka učení
Vývoj	Ekosystém balíčků
	Ekosystém
	Flexibilita
Komunita	Velikost komunity
	Aktivita komunity

Tabulka 4 - Kritéria hodnocení frameworků

4.3 Zvolené hodnocení

V této části práce nalezneme, podle čeho byla kritéria zvolena. Hodnotil se podle stejných poznatků v daných kritériích. Měřeno bylo během jednoho dne a pokud se data změnili, měřilo se znovu. Této části práce se již pracovalo se slovem „poznatek“. A to z toho důvodu, aby rozdíl měření byl pochopitelnější.

Důležité bylo porovnat charakter informací to zejména u poznatků kvantitativního, které je přirozeněji hodnotitelné. To neplatí však kvalitativního poznatků. Jeho charakter hodnocení byl slovní a musel se udatného pádný důvod tohoto hodnocení.

4.3.1 Stupnice bodování

Prvním krokem byla definice stupnice. Jak bylo řečeno v předešlém odstavci, byly definované stupnice kvantitativní a kvalitativní.

Stupnice:

- kvalitativní,
- kvantitativní.

Celkový počet bodů se vždy lišil. Důvodem byl odsouzení počet poznatků.

Stupnice kvantitativní

Stupnice ukazuje procentuální rozdíl. Borová ní probíjel od nejlepší hodnoty poznatků. Maximální dosažený počet bodů byl 10 a to však pokud byla hodnota nejlepší mezi ostatními. Zbylé hodnocení bylo podle bodu v tabulce. (Tabulka 5)

Rozdíl od maximálně vhodné varianty udána v %	Bodování
< 0; 10)	10
< 10; 20)	9
< 20; 30)	8
< 30; 40)	7
< 40; 50)	6
< 50; 60)	5
< 60; 70)	4
< 70; 80)	3
< 80; 90)	2
< 90; 100)	1

Tabulka 5 – Stupnice kvantitativního bodování

Pokud se jednalo o minimalizační povahu poznatku, kdy nejmenší hodnota byla nejlepší, rozdíly používaly vzorec:

$$x'_{ij} = 1 - \frac{min_i(x_{ij})}{x_{ij}}$$

Vzorec 1 - Minimalizační vzorec

Vzorce s hodnotou xij dané varianty a $mini(xij)$ je hodnotou mezi variantami minimálními. Maximalizační povahu poznatku, nejlepšího výsledku, použil vzorec:

$$x'_{ij} = 1 - \frac{x_{ij}}{max_i(x_{ij})}$$

Vzorec 2 - Maximalizační vzorec

Hodnota vzorce v *xij* je poznatek varianty a *maxi(xij)* značí vždy maximální hodnotu poznatku u vybraných frameworků.

Kvalitativní stupnice

Kvalitativní bodovací stupnice byla jiná než kvantitativní, a to že jsem nebodovalo procentuálně. Mezi nevýhody kvalitativní stupnice, patřilo to, že se na ní mohlo nahlížet jinak a to hlavně subjektivně.

Názory byly brány od senior vývojářů, odborníků z praxe, komunity a dle vlastních zkušeností. Každá komunita nahlížela na poznatky odlišný názorem. Stupnici nalezneme v tabulce (Tabulka 6).

Rozdíl od nejvíce vyhovující varianty	Bodování
Stejně vyhovující	10
	9
	8
	7
	6
	5
	4
	3
	2
Nejméně vyhovující	1

Tabulka 6 - Stupnice kvalitativního bodování

4.3.2 Zhodnocení dle kritérií

U kvalitativního hodnocení se využívali informace nalezené u komunit na webových portálech. Primární hodnocení dávali senior vývojáři. Tito vývojáři odpovídali ochotně na dotazy.

Manipulace DOM

Je interakce uživatele (manipulace) s prvky na internetové stránce. U jednostránkových SPA se stránky nerenderují znovu jako v případě MPA. A to hlavně

při ukládání dat do databáze a zpětné zobrazeni na UI uživateli. Z toho vyplývá, že výkon je velice důležitý, a to jak u frameworku, tak i knihovny. Nástroje, k získání kritérií, byl použit *js-framework-benchmark* a to nejen kvůli tomu, že můžeme změřit implementaci dle předem definovaných metrik ale i kvůli tomu, že každá komunita přidává svoji implementaci aplikace tohoto nástroje v určitém frameworku. (viz Příloha 1, 4, 5, 6)

Měření:

- vytvoření 1000 řádků tabulky po načtení stránky,
- záměny vytvořených řádků tabulky za 1000 nových,
- dílčí aktualizaci každého desátého prvku skrz 10 000 řádky tabulky,
- výběru a barevnému vyznačení po kliknutí na označený řádek tabulky,
- výměně dvou řádků mezi 1000 řádků v tabulce,
- odebrání jednoho řádku z 1000 řádků v tabulce
- vytvoření najednou 10 000 řádků,
- přidání 1000 řádků k již stávající tabulce s 10 000 řádků,

odebrání 10 000 řádků najednou. (viz Příloha 1, 4, 5, 6)

Výsledky (Tabulka 7) (viz Příloha 1, 4, 5, 6):

Manipulace s DOM	Angular	React	Vue
Vytvoření 1000 řádků	10	10	10
Substituce 1000 řádků	10	10	10
Částečná aktualizace mezi 10 000 řádky	9	10	9
Zvýraznění řádku	10	10	10
Prohození dvou řádků v tabulce mezi 1000 řádky	10	10	10
Odstranění řádku z 1000 řádků	10	10	10
Vytvoření 10 000 řádků najednou	9	10	9
Přidání 1000 řádků z 10 000 řádkům	10	10	10
Odstranění 10 000 řádků najednou	9	10	10
Σ (max 90)	**87**	**90**	**88**

Tabulka 7 - Hodnocení – Manipulace s DOM

Startovací metriky

Jak již víme u SPA se stránka vyrenderuje jenom jednou. Pro uživatele, kteří používají svoje data není tato funkcionalita vhodná, a to hlavně pro mobilní zařízení. A to je právě důležité pro hodnocení metrik, pokud je aplikace většího vzrůstu stahování dat trvá déle. Pokud uživatel načítá aplikaci neměl by opouštět tuto webovou stránku, neboť se může stát, že aplikace spadne nebo proběhne timeout, v lepším případě se bude načítat velice dlouho. Touto chybou jsou uživatele dosti odrazení od používání dané webové stránky. Tyto metriky primárně definuje Google. Díky *Lighthousu* se dají tyto metriky ručně změnit. Některé metriky dokáže zpracovat js-framework-benchmark. (viz Příloha 1, 4, 5, 6)

Měření:

- spouštěcí doby skriptu,
- síťoví přenos,
- interaktivita (TTI).

Jedna z mnoha *Lighthouse* funkcionalit je ta, že dokáže napovědět, jak optimalizovat daný web, aby pracoval rychleji. Proto bereme tuto funkci minutu za velice důležitou. Další funkcionalitou spuštění skriptu. Tu opět definuje Google. U této metriky vždy vyhodnocováno rychlost skriptu v milisekundách. V neposlední řadě můžeme *Lighthouse* nástroje měřit i síťový přenos. Měříme spotřebu odesílání zdrojů.

Bodování (Tabulka 8):

Startovací metriky	Angular	React	Vue
Spouštěcí doba skriptu	8	10	9
Síťoví přenos	8	9	10
Interaktivita (TTI)	8	9	10
Σ (max 30)	**24**	**28**	**29**

Tabulka 8 - Hodnocení – Spouštěcí metriky

Alokace paměti

Alokace paměti je další hledisko, ke kterému využíváme *Lighthouse*. Měření v příloze (viz Příloha 1, 4, 5, 6) U tohoto hlediska se měří zdrojová zátěž mobilního telefonu a počítače.

Měření zátěže:

- kliku k vytvoření 1000 řádků pětkrát za sebou,
- vytvoření a mazání 1000 řádků pětkrát za sebou,
- načítání stránky,
- přidání 1000 řádků,
- kliku k aktualizaci desátého řádku pětkrát za sebou.

Bodování (Tabulka 9):

Alokace paměti frameworku	Angular	React	Vue
Zátěž při vytváření 1000 řádků(5x)	8	10	9
Zatížení při vytváření a mazání 1000 řádků (5x)	8	10	9
Načítání stránky	8	9	10
Přidání 1000 řádků	8	9	10
Zatížení při částečné aktualizaci(5x)	8	10	9
Σ (max 50)	**40**	**48**	**47**

Tabulka 9 - Hodnocení – Alokace paměti

Velikost komunity

Jak již víme z předešlých kapitola důležitá pro vývojáře je komunita. Výhodou velké komunity je jen dokumentace ale i výukové materiály, které najdeme k dispozici na YouTube případně na webu UDEMY. Na základě této práce byly prozkoumány různé weby a místnosti pro komunikaci vývojářů kde bylo možné porovnat frameworky mezi sebou. Jednalo se o komunikativní kanály podle komunit.

GitHub je portál kde vývojáři sdílí svoje repozitáře s kódem a v rámci kontribuce je upravují. Na tomto portál lze zjistit velikost komunity. V prvním případě jde zjistit oblíbenost frameworku na základě počtu hvězdiček ale jenom na základě určité míry. Dále je možné zjistit kolik vývojářů tento framework používá na základě stahování. Je také zjistit kolik vývojářů se podílí na kontribuci daného frameworku.

Dále jsou známe komunikativní kanály jako je třeba Twitter anebo *Reddit*. Jsou to sociální sítě. *Reddit* je sociální síť kde se mohou zakládat různé subreddity[2] a slouží jako fóra. Díky tento funkcionalitě bylo mnohem jednodušší zjistit kolik odběratelů využívá *subreddit*[2]. Pro vybraný Framework se vybíral *supereddit*[2] který měl nejvíc odběratelů. S nejvíce využívající komunikační portál je Stack Overflow. Tento portál využívají vývojáři pro komunikaci a odpovědi na jejich otázky. Dalším známým nástrojem pro komunikaci je *Discrod*[3]. Dnes ho již využívá spoustu vývojářů, ale pokud hledáme vývojářskou komunitu, ta využívá nástroj *Gitter*[4]. Oba nástroje slouží jako komunikační místnosti. Komunita *Angularu* využívá *Gitter*[4]. *Discrod*[3] využívají komunity *React* a *Vue*.

[2]Subreddit*: r/reactjs, r/vuejs, r/angular*

[3]Discord (Vue, Reac,): *https://discord.com/invite/Reactiflux, https://discord.com/invite/HBherRA*

[4]Gitter (Angular)*: https://gitter.im/angular/angular*

Hodnocení:

- hodnocení GitHubu,
- počet přispěvatelů na GitHub,
- počet využívajících na GitHubu,
- počet vyřešených otázek na Stack Overflow,
- počet členů v chatovacích místnostech,
- počet sledujících na Twitteru,
- počet odběratelů na *Redditu*.

Bodování (Tabulka 10) :

Velikost komunity frameworku	Angular	React	Vue
Hodnocení GitHubu	5	10	3
Počet přispěvatelů na GitHub	8	10	6
Počet využívajících na GitHubu	8	10	6
Počet vyřešených otázek na Stack Overflow	10	10	10
Počet členů v chatovací místnostech	8	10	8
Počet sledujících na Twitteru	8	10	8
Počet odběratelů na Redditu	8	10	8
Σ (max 70)	**55**	**70**	**49**

Tabulka 10 - Hodnocení – Velikost komunity frameworku

Aktivita komunity

Aby mohl být framework považovaný za funkční a existenčně kompletní, je potřeba zohlednit rovněž aktivitu v rámci komunity, která ho využívá. Od aktivity se odvíjí schopnost určit, zda je framework vhodný k použití s ohledem na jeho využití a budoucí podporu. V případě, kdy bude framework nespolehlivý, nelogicky napsaný apod., nebude využíván a komunita kolem něj bude výrazně malá a neaktivní. Ve prospěch open-source projektů hraje roli výraznější vývoj různých knihoven a nástrojů. To se také pozitivně projeví na křivce učení. Ze strany hodnocení je aktivita, podobně jako velikost, výrazněji preferována.

Jak je známo, aktivní komunita dokáže pomáhat určitým vývojářům nejen velkým množstvím studijním materiálům. Výhodou je, že kromě čtení dokumentace mohou pomáhat například tím, že odpovídají na mnohem více specifikované dotazy. Velikou výhodou je, že na odpověď se nečeká tak dlouho aby vývojář nemohl pokračovat dále v práci. Často se stává, že dokumentace není dostačující, a je tím pádem možné využití komunikačních kanálů, kde se odpovědi dostane.

Nejdůležitější aspekt aktivity je detekce možných chyb. Vývojáři velmi často plánují využít framework v rozličných oblastech. V rámci hodnocení tohoto kritéria došlo nejprve k analýze, jak by bylo možné efektivně a správně aktivitu vyhodnotit. V rámci této analýzy byly postupně vyjádřeny konkrétní kanály, které jsou pro určení aktivity relevantní. Pro *npm* platí, že aktivitu je možné měřit na základě toho, kolikrát byl daný framework stažen. Z počtu stažení lze také do jisté míry vyjádřit, jak moc je daný framework používán. Z počtu stažení nicméně nelze vyjádřit to, k jaké aktivitě dochází na různých informačních a podpůrných zdrojích, viz, například Stack Overflow.

Stack Overflow je nej větším zdrojem informací pro mnoho vývojářů. Měření aktivity v této stránce bylo provedeno tak, že byly zkoumány počty příspěvků, ke kterým zároveň existovala odpověď. Pro účely této práce bylo nastaveno omezení časového rozsahu pro výběr odpovědí a to tak, aby byly vybrány pouze takové příspěvky, které splnily podmínku na publikování 1 měsíc do minulosti. Tento způsob byl zvolen z toho důvodu, že takto vybrané informace dle autora nejlépe ukážou aktuální míru aktivity i s ohledem na to, že informace byly získávány z různých komunit. Mezi další zdroje patří síť *Reddit*[5].

Hodnocení:

- počet stažení za týden na *npm*,
- počet komentářů za jeden den na *Redditu*,
- počet vyřešených otázek za poslední měsíc na Stack Overflow,
- počet příspěvků na *Redditu* za jeden den.

[5]*Dostupné na https://subredditstats.com (Subreddit Stats, 2022)*

Výsledky hodnocení tohoto kritéria (Tabulka 11):

Aktivita komunity frameworku	Angular	React	Vue
Počet stažení na npm (týden)	1	10	2
Počet komentářů na Redditu (den)	8	10	8
Počet vyřešených otázek za (měsíc) na Stack Overflow	8	10	8
Počet příspěvků na Redditu (den)	8	10	8
Σ (max 40)	**25**	**40**	**26**

Tabulka 11 - Hodnocení – Aktivita komunity

Dokumentace

Dokumentaci bereme jako důležitý materiál pro vývojáře, ze kterého čerpají potřebné informace o frameworku. Kvalita dokumentace často znamená vyšší použitelnosti frameworku. Co se týká obsahu, obecným předpokladem je, že v ní budou obsažené veškeré informace a postupy, které uživatel dokáže využít tak, aby pochopil podstatu a ulehčilo mu to samotnou práci. Aby bylo možné provést komparaci mezi jednotlivými frameworky, byly pro účely práce vybrány takové oblasti, které lze mezi sebou nějakým způsobem porovnat. Zároveň do porovnání nebyly zahrnuty žádné materiály, které přímo nesouvisely s oficiální dokumentací.

Nejdůležitější je to, aby dokumentace obsahovala vše, co je třeba k použití určitého frameworku. Měla by obsahovat dané informace o prvním nastartování a jak jej docílit. Je tedy velice důležitou náplní daného frameworku.

S ohledem na náplň dokumentace je vždy vhodné také dovolit určitou kooperaci z řad komunity, a to například formou spolupodílení se na GitHubu. Podstatou tohoto spolupodílení je minimalizovat možné nepochopení dokumentace způsobené tím, že autor určitou oblast nedostatečně popíše, či jí popíše stylem, který nemusí následným čtenářům dávat jednoznačný smysl. Jelikož zásadním prvkem při použití frameworku je tzv. API reference, bývá tento doplněk součástí dokumentace, kdy dochází k jejímu automatickému vygenerování a je součástí kódu. Dochází ke generování primárně určitých částí, které jsou k užití pro uživatele. Protože je zdokumentovaný samotný kód, tak je často reference na API bránu označována za ne tolik potřebnou. Jako hlavní pozitivum API reference, by se dala zmínit velice nízká složitost vyhledávání. Ze stran vývojářů je ovšem obvykle preferováno

neprohlížet přímo zdrojový kód. Tím pádem je spíše vhodné, aby si každý vývojář vybral sám, co preferuje.

Jestliže v dokumentaci nalezneme prvotní postupy a další užitečné informace, znamená to také efektivnější použitelnost. V návodech jsou také velmi často v rámci postupů přiblíženy možnosti a funkcionality určitého frameworku. Vývojáři vůbec nejsou nuceni vyhledávat si návody, jak psát první aplikaci. Ale existují ti, kteří to využívají. A to znamená, že pro větší vypovídající hodnotu této práce došlo tedy k zohlednění i těchto případů. Co se týká samotných návodů, je obecně doporučováno přikládat i *cheatsheet*.

Ten obsahuje základní úryvky kódu, které jsou použitelné nejenom nováčkům ale také současným uživatelům, a to zejména proto, protože je lze jednoduše zkopírovat a využít na potřebném místě. Logicky lze předpokládat, že typický uživatel si nepřečte dokumentaci v jednom okamžiku celou a dále se k ní již nikdy nevrací. Naopak je běžný stav, kdy bývá otevírána častěji ať už v rámci ověření si informací, či dohledání detailnějších vlastností. Z tohoto důvodu se považuje za standard, aby byla dokumentace umístěná na místě, kde je možné k ní jednoduše přistupovat a vyhledávat v ní (typicky se jedná o webové stránky).

Mimo jiné se v dokumentaci obvykle nachází také různé interaktivní ukázky, které lze využít jako podklad pro činnost vývojáře. Příklady jsou pozitivní především proto, protože si uživatel může daný framework vyzkoušet a není nutné pro tyto případy vymýšlet složité ukázky. Další výhodou je, že není nutné si framework přímo implementovat.

V praxi je poměrně obvyklé, že uživatelé nečtou dokumentaci celou, ale vybírají pouze úryvky, které jsou pro ně zajímavé. Mezi nejčastější důvody tohoto chování patří především fakt, že kompletní dokumentace je mnohdy velmi rozsáhlá. Z tohoto důvodu, jak již bylo zmíněno, je obecně preferované mít k dispozici strukturované vyhledávání, díky kterému si uživatel vyhledá konkrétní informace a není nucen procházet kompletní dokumentaci.

Dalším měřítkem, které je možné vzít v potaz je i lokalizace dokumentace do rodného jazyka. Ačkoliv je obecným zvykem mít dokumentaci psanou v obecně používaném jazyce, tzn. Většina dokumentací je psaná v angličtině, pokud by se jednalo o složitější funkci, může být pro uživatele lépe pochopitelné, pokud bude popsána v jeho rodné řeči. Toto zároveň poskytuje výhodu pro uživatele, kteří nemají takové znalosti jazyka, aby zvládli funkci pochopit v angličtině.

Hodnocení:

- náplň dokumentace,
- kontribuce,
- *cheatsheet*,
- interaktivní příklady,
- první kroky a návody,
- vyhledávání,
- API reference,
- a dostupné jazyky.

Zhodnocení dokumentace (Tabulka 12):

Dokumentace frameworku	Angular	React	Vue
Náplň dokumentace	10	10	8
Kontribuce	10	10	8
Cheatsheet	10	10	8
Interaktivní příklady	10	9	9
První kroky a návody	10	10	10
Vyhledávání	10	10	10
API reference	10	10	10
Dostupné jazyky	8	10	5
Σ (max 80)	**78**	**79**	**68**

Tabulka 12 - Hodnocení – Dokumentace frameworku

Křivka učení

Toto kritérium v sobě může zahrnovat různorodé ukazatele. Často se odvíjí od toho, pro jaké způsoby užití je daný framework koncipován a také podle toho, jak moc složitá je práce s ním. Toto ale není jediný ukazatel. Mezi další lze považovat i vliv ostatních uživatelů, tzn. Jak moc např. uživatelé publikují určité návody pro práci s frameworkem nebo jak moc aktivní jsou z pohledu konzultace při řešení problémů ostatních vývojářů.

Křivka učení je pak také poměrně ovlivněna tím, zda si uživatel vystačí se samotným frameworkem nebo musí pro náročnější implementace využívat různé dodatečné knihovny.

Vzhledem k tomu, že primární směr této práce spočívá v porovnání frameworku samotného (tzn. Bez dalších složitějších doplňků apod.), nebyl například zohledněn ukazatel týkající se nabídky materiálů pro studium vytvořených komunitou, a to především z toho důvodu, že se nejedná o ukazatel, který by šel přímo porovnávat mezi sebou pro jednotlivé frameworky.

Při definování ukazatelů byl určen předpoklad, kdy počítáme, že uživatel se řadí spíše k nováčkům v oboru, a ne k seniornímu vývojáři. Avšak nelze považovat úplného nováčka. Výchozí persona je tedy uživatel, který zvládá HTML, CSS a JS a zároveň jde o uživatele, který nemá žádné zkušenosti s frameworky.

V souvislosti s naposledy zmíněným předpokladem je ale nutné dodat, že před použitím daného frameworku se dá předpokládat, že si o něm uživatel nějaké údaje předem dohledá. V dnešní době spousty frameworků nabízí nad stavbu tzv. *Typescript*. Tato nadstavba byla také součástí hodnocení a to proto, pokud framework vyžaduje nadstavbu je po uživatele vyžadováno, aby se s touto nadstavbu seznámil. Častý problém je to právě pro začínající vývojáře.

Všechny frameworky porovnávané v této práci nabízí tvorbu aplikací pomocí komponent, ze kterých se tyto aplikace následně skládají. Dalším kritériem je tedy porovnání složitosti práce s těmito komponentami. Složitostí je myšlena především složitost jejich tvorby, schopnost jejich vykreslování a také to, jakým způsobem mohou být následně využité v aplikaci.

Některé frameworky podporují také *dependenci injection*. Pro vysvětlení se jedné o nástroj/techniku, se kterou se lze nejčastěji setkat v případně objektově orientovaných jazyků. V případě uživatelů, kteří tuto techniku ovládají jde o velké usnadnění práce. Nicméně pro uživatele, kteří jí neovládají může jít naopak o poměrně složitou záležitost. I pro mírnou nevýhodu pro začínající uživatele bylo kritérium využití této techniky daným frameworkem rovněž zahrnuto do porovnání.

K dalším běžně používaným technikám patří práce s data-binding. Jde o vazbu, která se využívá k provázání spotřebitelských dat s daty poskytovatelskými. Tzn. Že tuto vazbu lze chápat jako spojení, které zajistí, že data mezi těmito dvěma stranami budou vždy synchronizovaná. Existuje více způsobů využití této vazby, a to především v závislosti na tom, kdo bude data měnit. Pokud jde o data, která jsou modifikována pouze jednou stranu, lze využít tzn. Jednostrannou vazbu. Tím, že jsou data měněna pouze jednou stranou, je zajištěna větší stabilita aplikace. Při oboustranné vazbě, jak z názvu vyplývá, jsou data měněna oběma stranami, což klade vyšší nároky na celkovou synchronizaci dat.

Vzhledem k tomu, že konkrétní řešení závisí na tom, jaký účel bude daná aplikace mít, bylo hodnoceno i to, zda framework podporuje možnost výběru, tzn. Zda si uživatel může zvolit, kterým způsobem bude synchronizaci řešit.

Frameworky také nabízí podporu pro práci v tzv. asynchronním režimu. Tento způsob, jak z názvu vypovídá, umožňuje uživateli využívat běhu více operací naráz. Jedná se především o asynchronní zasílání dat, kdy není nutné čekat na doběhnutí prvního volání, ale je zároveň spuštěno i další volání. Následně po získání odpovědi na první volání už může běžet další proces i když všechna volání ještě nedoběhla. Z pohledu uživatele je tak aplikace rychlejší. Kritériem pro hodnocení tohoto způsobu byla složitost jeho implementace v konkrétní frameworku.

Jelikož ne vždy je reálné využívat pro komplexnější aplikace pouze samotný framework, byla zahrnuta i oblast, která se zaměřuje na to, do jaké míry lze využít podpory ze stran knihoven, které pocházejí od třetí strany. Kromě složitosti implementace těchto knihoven byla hodnocena i náročnost pro naučení se s danou knihovnou pracovat.

Hodnoceno bylo na základě těchto poznatků:

- psaní komponent,
- způsob použití komponent,
- psaní nebo vykreslování šablon,
- práce s *TypeScript*,
- znalosti *dependency injection*,
- práce s vazbou dat (data binding),

- práce s asynchronním kódem,
- míra potřebné práce s knihovnami třetích stran.

Bodování křivky učení (Tabulka 13):

Křivka učení frameworku	Angular	React	Vue
Psaní komponent	8	9	10
Použití komponent	8	10	10
Psaní a vykreslování šablon	10	9	10
Práce s TypeScript	8	10	10
Techniky Dependency Injection	10	8	8
Práce s Data Binding	10	8	8
Práce s asynchronním kódem	10	10	10
Míra potřebné práce s knihovnami třetích stran	10	9	10
Σ (max 80)	**74**	**73**	**76**

Tabulka 13 - Hodnocení – Křivka učení

Dostupné knihovny

Běžná praxe ukazuje, že mnoho uživatelů/vývojářů si práci poměrně často ulehčují tím, že pro svou práci využívají kód, který vytvořil už někdo před nimi. V tomto případě se nemusí nutně jednat o pohodlnost vývojáře, ale má to i praktické důvody. Především ten, že není nutné kód pro stejnou funkcionalitu opětovně vymýšlet, což vede k ulehčení práce.

V rámci tohoto daného kritéria bylo hodnoceno jen to, kolik dosažitelných knihoven třetích stran bývá pro zvolený framework. A to v okruhu balíčků *npm*.

Hodnoceno:

- počtu balíčků *npm*.

Hodnocení kritéria (Tabulka 14):

Dostupné knihovny	Angular	React	Vue
Počet balíčků npm	8	10	5
Σ (max 10)	**8**	**10**	**5**

Tabulka 14 - Hodnocení – Dostupné knihovny

Ekosystém

Ekosystém tvoří nástroje a knihovny frameworku. Je několik druhů ekosystému. Stabilitu aplikace udává kvalita ekosystému. Často je potřeba aplikaci optimalizovat. K tomu se využívá *tree-shaking* technika. Častý problém bývá ten, že Framework za nějaký čas vyžaduje aktualizace. Pokud majitel frameworku neaktualizuje knihovnu na nejnovější verze, může nastat problém. Důležitá část frameworku je také DOM, neboli stav aplikace, routování, práce s formuláři a validacemi.

Hodnocení:

- správa stavu,
- manipulace s DOM,
- zpracování formulářů a validace,
- routování,
- nástroje k testování.

Výsledky hodnocení (Tabulka 15) :

Ekosystém frameworku	Angular	React	Vue
Správa stavu (state)	10	9	8
Manipulace s DOM	8	9	10
Validace formulářů a zpracování	10	9	8
Routování	10	9	8
Nástroje k testování	10	10	10
Σ (max 50)	**48**	**46**	**44**

Tabulka 15 - Hodnocení – Ekosystém

Flexibilita

Flexibilitu chápeme jako svobodný vývoj ve vybraném frameworku. Je dobré poznamenat, že na flexibilitu lze pohlížet pozitivně ale také negativně. A to především kvůli tomu, že je zde velkou roli hrají preference jednotlivých uživatelů. Lze se totiž setkat s jedinci, kteří kladou velký důraz na to, aby existovala konkrétní pravidla a postupy, kterými

lze nějaký konkrétní problém řešit. Oproti tomu pozitivní flexibilita spočívá v tom, že existuje více možných řešení, která nejsou striktně daná. Právě na druhý druh flexibility zkoumá tato práce. Tento druh flexibility je dnes vyžadován především v aplikacích, které svému uživatele nabízí určitou formu interaktivity a jsou zaměřené především na to, aby poskytly uživateli kladný zážitek.

V dnešní době není výjimkou situace, kdy dochází k modernizaci již existujícího webu. Ten ale nemusí být nutně implementovaný pomocí JS jazyka a tím pádem může mít problémy s interaktivitou. Proto se flexibilita dá pojmout i jako, hledisko zavádění frameworků do existujících projektů s cílem využití určité části frameworku, jako například na formuláře anebo na postupné aktualizaci kompletní aplikace. Jako další hodnocený prvek bylo vybráno to, do jaké míry framework vyžaduje držet se konkrétních postupů pro využití nástrojů či určitých struktur. Tím, že některé frameworky vychází z konkrétní architektury, dává hodnocení tohoto kritéria smysl. Nicméně ne vždy je možné tento pohled vztáhnout na všechny frameworky, jelikož jsou některé v tomto ohledu volnější.

Často využívanou technikou webových aplikací je Server Side Rendering, který slouží k tvorbě dynamického obsahu webových stránek. Lze ho použít například pro optimalizaci vyhledávačů.

Hodnocení:

- striktnost pravidel vývoje,
- možnosti data bindingu,
- možnost zavedení do existujícího projektu,
- možnosti vykreslování na straně serveru,
- možnosti psaní komponent,
- možnosti šablonování a vykreslování,
- možnost multiplatformního vývoje,
- možnosti stylování.

Výsledky hodnocení (Tabulka 16)

Flexibilita frameworku	Angular	React	Vue
Striktní pravidla vývoje	7	10	7
Data Binding	10	8	10
Možnost zavést do existujícího projektu	7	8	10
Možnosti vykreslování na straně serveru	10	10	10
Psaní komponent	8	10	8
Psaní šablon/vykreslování	10	8	10
Možnost multiplatformního vývoje	10	10	8
Stylování	8	10	8
Σ (max 80)	**70**	**74**	**71**

Tabulka 16 - Hodnocení – Flexibilita

4.3.3 Model vícekriteriální analýzy variant

Metodou vícekriteriální analýzy variant bylo realizováno finální hodnocení. Následně byla využita metoda Saatyho a metoda váženého součtu. Váhy byly získány Saatyho metodou následně goniometrický průměr a váha pro kritérium. Celkový užitek byl vypočten metro váženého součtu. Pomocí Saatyho metody byla vymodelována Saatyho matice a každé kritérium bylo ohodnoceno body, a to na základě průzkumu.

Kritéria (v Tabulce 17).

Kritéria		Maximální počet bodů
Manipulace s DOM	D1	90
Startovací metriky	D2	30
Alokace paměti	D3	50
Velikost komunity	C1	70
Aktivita komunity	C2	40
Dokumentace	P1	80
Křivka učení	P2	80
Ekosystém balíčků	E1	10
Ekosystém	E2	50
Flexibilita	E3	80

Tabulka 17 - Kritéria hodnocení frameworku

Saatyho metoda

Nejprve byli osloveni senior vývojáři pro shodu preferencí. Na základě toho byli získané výsledky k řešení. Ty měli nejnižší konzistenční poměr. V tabulce (Tabulka 18) je Saatyho matice.

	D1	D2	D3	C1	C2	P1	P2	E1	E2	E3
D1	1	1/4	1/3	1/4	1/5	1/5	1/6	1/5	1/6	1/5
D2	4	1	4	1/5	1/5	1/6	1/6	1/5	1/5	1/5
D3	3	1/4	1	1/4	1/4	1/4	1/5	1/5	1/5	1/5
C1	4	5	4	1	1/3	1/3	1/3	1	1	2
C2	5	5	4	3	1	1/2	1/3	1	1	2
P1	5	6	4	3	2	1	1/3	1	1	3
P2	6	6	5	3	3	3	1	1	1	2
E1	5	5	5	1	1	1	1	1	1	3
E2	6	5	5	1	1	1	1	1	1	1
E3	5	5	5	1/2	1/2	1/3	1/2	1/3	1	1

Tabulka 18 - Tabulka Saatyho metody kritérií hodnocení frameworků

Dalším krokem bylo nutné provést konzistenci Saatyho Matice. Vyjádření proběhlo na základě indexu konzistence. Po tomto výpočtu bylo možné získat maximální číslo matice. K vyjádření byl využit online nástroj Wolfram Alpha. Výsledek se rovnal 11,7288. Tato hodnota se využila ke konečnému výpočtu indexu:

$$CI = \frac{11{,}7288 - 10}{10 - 1} \approx 0{,}1921$$

Vzorec 3 - Index konzistence

Hodnotu z indexu lze využít v dalších výpočtech, jako například k poměru konzistence. Dalším krokem byl výpočet náhodného indexu. Na základě počtu kritérií

je tento index pevně daný. V této práci bylo využito 10 kritérií a jejich RI hodnota se rovnala 1,49. Z výsledných indexů lze vypočítat poměr konzistence:

$$CR = \frac{0{,}1921}{1{,}49} \approx 0{,}1289$$

Vzorec 4 - Poměr konzistence

Výsledek výpočtu se rovnal 0,1289. Na základě výpočtu a jeho hodnoty lze považovat vypočtenou hodnotu jako validní a konzistentní pro Saatyho metodu k dalšímu využití. Dalším krokem byl výpočet vah kritérií. Pro každý řádek kritéria byl vypočten průměr a suma těchto průměrů byla využita k dopočtu požadovaných vah. (Tabulka 19).

	Goniometrický průměr	Váhy kritéria
D1	4,2514	0,3175
D2	2,7373	0,2044
D3	2,0518	0,1532
C1	0,2786	0,0208
C2	0,3717	0,0277
P1	1,3443	0,1004
P2	1,0077	0,0752
E1	0,6607	0,0493
E2	0,6834	0,0510
E3	0,6607	0,0493
Σ	13,3874	1,000

Tabulka 19 - Goniometrický průměr frameworků a vypočtené váhy kritérií

Z výsledné hodnoty vah bylo možné odvodit, kterou křivku lze považovat za křivku s největším vlivem. V tomto případě šlo o křivku reprezentující učení a dokumentaci, které se zaměřuje především na použitelnost konkrétního frameworků. Další v pořadí vzešla kritéria reprezentující aspekt výkonu, konkrétně ekosystém a knihovny. Ty se využívají k usnadnění práce s frameworky. Rovněž vlivnou, se ukázala být křivka reprezentující komunitu z pohledu jak aktivity, tak také její velikosti.

Fakt, že toto kritérium vyšlo jako jedno v nejvíce vlivných dává smysl, neboť právě komunita je základ pro to, aby byl framework udržovaný, rozvíjený a například k němu mohli vznikat i nové knihovny. Za nejméně vlivnou byl identifikován výkon. Výkonnost je tedy na jednu stranu vhodné znát, nicméně obecně platí, že tato veličina může být do značné míry ovlivněna tím, jak moc je vývojář schopný s aplikací pracovat a zda jí dokáže správně optimalizovat. Toto kritérium by tedy bylo vhodné mírně upozadit. Co je ale nutné zmínit je to, že jednotlivé váhy mohou být poznamenány subjektivním pocitem. Pro výpočet vah byl osloven uživatel z praxe, který autorovi asistoval a byl požádán, aby provedl srovnání kritérií na základě svých zkušeností.

Metoda váženého součtu

Další využitou metodou byla metoda váženého součtu. Tato metoda je založena tím, že určí dílčí užitky zvolených variant. To znamená, že je poté možné určit pro každou variantu jejich celkový užitek. Jako první bylo potřeba zjistit dílčí užitky zvolených kritérií.

Díky tomu, že všechna zvolená kritéria byla zvolena tak, aby bylo možné zvolit metodu maximalizačního rázu, nebylo nikterak potřeba provádět ve výpočtech další nadbytečné zásahy z důvodu převodů a podobně. V tabulce níže (Tabulka 20) je uvedeno bodové hodnocení pro jednotlivé varianty a kritéria.

	D1	D2	D3	C1	C2	P1	P2	E1	E2	E3
Angular	87	24	40	55	25	78	74	8	48	70
React	90	28	48	70	40	79	73	10	46	74
Vue	88	29	47	49	26	68	76	5	44	71

Tabulka 20 - Hodnoty kritérií variant frameworků

Následně došlo k určení vektorů pro ideální a bazální varianty. Výsledné varianty jsou uvedeny v tabulce níže (Tabulka 21).

	D1	D2	D3	C1	C2	P1	P2	E1	E2	E3
D	87	24	40	49	25	68	73	5	44	70
H	90	29	48	70	40	79	76	10	48	74

Tabulka 21 - Bazální a ideální varianty hodnocení frameworků

V neposlední řadě bylo provedeno vyjádření užitku všech variant. Pomocí metody Vážená součtu byly vyjádřeny užitky a váhy byly vypočteny Saatyho metodou. Nakonec byl vyjádřen skalární součin všech užitků. Výsledný užitek pro křivku = 6 a variantu *Vue* = 3 byl vypočten:

$$u_{3,6} = \frac{78 - 68}{79 - 68} = 0{,}91$$

Vzorec 5 - Normalizace hodnot

Podobným způsobem došlo k vypočítání ostatních hodnot. Výpočty uvedené v tabulce vyjadřují dílčí užitky kritérií pro jednotlivé varianty.

	D1	D2	D3	C1	C2	P1	P2	E1	E2	E3
Angular	0	0	0	0,29	0	0,91	0,33	0,60	1	0
React	1	0,8	1	1	1	1	0	1	0,50	1
Vue	0,33	1	0,88	0	0,07	0	1	0	0	0,25

Tabulka 22 - Normalizovaná matice frameworku

V poslední řadě byl vyjádřeny celkové užitky všech variant. Pomocí metody Vážená součtu byly vyjádřeny užitky a váhy byly vypočteny Saatyho metodou v předešlé části. Nakonec byl vyjádřen skalární součin všech užitků. Celkový užitek pro variantu *Vue* = 2 byl vypočítán následovně:

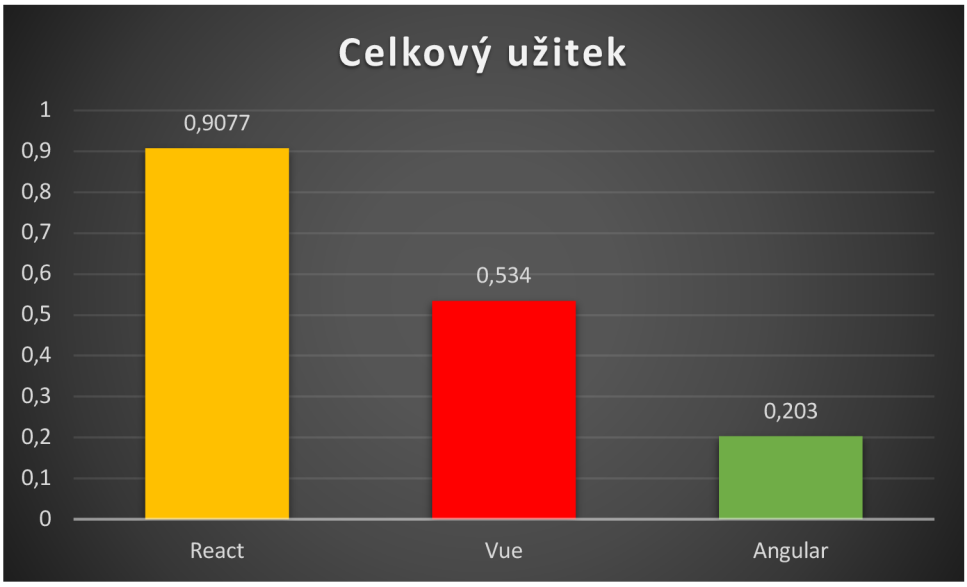
$$u(\text{D3}) = 0{,}33*0{,}32+1*0{,}20+0{,}88*0{,}15+0*0{,}02+0{,}07*0{,}03+0*0{,}1+1*0{,}08+0*0{,}05+0*0{,}05$$
$$+0{,}25*0{,}05 =\approx 0{,}53$$

Vzorec 6 - Skalární součin dílčích užitků

	Celkový užitek	Pořadí frameworku
React	0,9077	1.
Vue	0,5340	2.
Angular	0,2030	3.

Tabulka 23 - Výsledné hodnocení frameworků

Po výpočtu všech užitků se hodnoty výsledných hodnot seřadili do tabulky (Tabulka 23). Výsledek ukazuje míru učení a jiné možnosti vybraného frameworku. Jde o dostupnost knihoven a ekosystém frameworku. Jedny z hlavních kritérií byly aktivita a velikost komunity.



Graf 1 - Výsledné užitky variant hodnocení frameworků

Z finálních výsledků jednotlivých srovnání lze vyvodit fakt, že jako nejlepší může být považován *React*, který dosáhl užitku 0,91. Jako „nejhorší“ z testovaných se ukázal být *Angular*. U něj byl spočítán užitek na 0,20. Největším hybatelem ve výsledku *Angularu* byla křivka učení.

5 Výsledky a diskuse

Tato kapitola se věnuje vyhodnocení výsledků získaných během tvorby práce. Podstatou je tedy finální vyjádření jednotlivých frameworků, a to na základě vydefinovaných kritérií. Pro lepší pochopení rozdílů mezi některými variantami, bylo potřeba získat další informace.

V praktické části je použito podobného metodického postupu jako bylo použit v práci Ing. Pocara 2021, *Vývoj webových aplikací pomocí JavaScript frameworků*. Z výsledků prací lze predikovat trend v řešené oblasti Javascriptových frameworků.

5.1 Analýza zhodnocených vybraných frameworků

Co se týká výkonu, zde lze poměrně jasně říci, že mezi frameworky nedocházelo k výraznějším odlišnostem. Jedním z důvodů, proč tomu tak je, je především to, že výkon jako takový nelze přímo porovnávat a z velké částí závisí, na jakém druhu aplikace je framework využitý a jak je tato aplikace optimalizovaná. Právě optimalizaci lze zařadit mezi důležité kritérium, které by měl vývojář ovládat. Znatelnější rozdíly lze pozorovat u komunity. V tomto ohledu se nejlépe zařadil *React*. Další zkoumanou oblastí byla použitelnost, v rámci, které byla hodnocena i křivka učení. V této oblasti si nejhůře vedl *Angular*, který lze tedy považovat za, ze všech zkoumaných frameworků, nejsložitější na naučení. Opačná situace je u *Vue*, který byl naopak vyhodnocen jako nejsnazší. U zkoumání z pohledu vývoje se výsledky lišily pro každý konkrétní aspekt.

5.1.1 Manipulace s DOM

V této kategorii se z velké části všech měřených disciplín umístil *React*, co by jeden z rychlejších. Jediná oblast, ve které zaostával za *Vue*, bylo zaměňování dvou řádků v 1000 řádcích tabulky. S obdobným problémem s potýkal také *Angular*, který měl navíc problémy v situaci, kdy bylo vyžadováno provést promazání kompletně všech vyskytujících se prvků na stránce. Nejlepších výsledků dosáhl *Vue*. Ten nicméně oproti ostatním mírně ztrácel v situaci, kdy bylo vyžadováno zvýraznit barevně prvky na stránce.

5.1.2 Startovací metriky

U spouštěcích metrik si velmi obdobně vedly frameworky *React* a *Vue*. Z pohledu stahování zdrojů bylo možné pozorovat mírnější rozdíly mezi *Angularem* a *Reactem*.

Jak je možné vidět vypočtených výsledku lze pozorovat různé odlišnosti daných frameworku. Například neprospěch *Angular* u spouštění skriptu a doby kompilace. V těchto disciplínách délka trvání byla 140 ms. V jiných oblastech to byla doba 16 ms.

5.1.3 Alokace paměti

Výsledné hodnoty tohoto kritéria můžeme považovat za podobné. Nejméně bodů získal *Angular*. U *Reactu* a *Vue* byly viditelné pouze minimální rozdíly.

5.1.4 Velikost komunity

Největší komunitou vybraných frameworků je komunita *Reactu*. Z výsledku lze odvodit, že framework je velmi populární a využívaný, což lze také vidět na výsledcích výzkumu State of JS 2022 Rozsáhlá komunita *Reactu* zaručuje dostupnost velkého množství výukových materiálů i knihoven třetích stran.

Pokud budeme mluvit o velikosti komunity, lze s určitostí tvrdit, že nejmenší lze nalézt u *Angularu*. Rozdílnost lze najít především v oblasti různých místností určených pro chat. Ačkoliv v tomto případě je velikost ovlivněna také skutečností, že se komunita dělí na dvě části *AngularJS* a *Angular*. Vzhledem k tomu, že u *AnglularJS* došlo ke značnému přepracování do nové verze a tato verze je od původní znatelně odlišná, byly vyhodnocovány pouze údaje týkající se této verze.

Z pohledu finálního hodnocení se nejhůře umístil *Vue*. V otázce aktivních uživatelů (dle GitHub) lze v porovnání s *Reactem* hovořit dokonce o 97 % rozdílu. Podobně byla vyhodnocena také aktivita, které byla zkoumána dle webu Stack Overflow, kde se *Vue* rovněž umístil jako poslední z pohledu počtu přispívajících uživatelů. Po tomto zjištění byl proveden ještě detailní průzkum, na základě, kterého bylo zjištěno, že v porovnání mezi všemi testovanými frameworky, má *Vue* největší základnu aktivních uživatelů v Číně (kde

mají zbylé dva naopak zástupnost velmi malou). Ve prospěch mu hraje také fakt, že je využíván tamními giganty (např. *Xiaomi, Alibaba, Bilibili* apod.).

Vysvětlení tohoto velkého rozšíření v Číně je dáno pravděpodobně v největší části tím, že autor frameworku Číny pochází a tím pádem byl tento framework v Číně také poprvé představen. Dalším faktem, který mluví ve prospěch *Vue* je to, že zbylé dva frameworky (resp. Jejich vlastníci – Facebook a Google) mají platný zákaz využívání na území Číny. Tamní vývojáři tedy logicky musejí dát přednost *Vue*. Dalším nedostatkem z pohledu podpory komunity je fakt, že velké množství knihoven vytvořených právě čínskými vývojáři nabízí dokumentaci pouze v jejich rodném jazyce.

5.1.5 Aktivita komunity

Výsledek tohoto kritéria bylo již možné odhadnout s velikostí určitých komunit. Díky velikosti komunit bylo jasné, že *React* dle portálu Stack Overflow je nejvíce probíraný framework a také nejvíce stahovaný. Z tohoto důvodu holce brát za číslo jedna. Ostatní zbylé frameworky lze brát za podobné. V potaz bylo bráno také velikost komunity těchto frameworků, které jsou v aktuální verzi *Angularu*.

5.1.6 Dokumentace

Co se týká dokumentace, zde nebyly pozorovány větší odlišnosti. Za zmínku stojí rozdíly například u některých dodatečných funkcí. Jedná se zejména o jazykovou rozmanitost dokumentace. Obsah *React* dokumentace byl vyhodnocen jako dostatečný. Nicméně v určitých oblastech byl spíše stručnější, díky čemuž si uživatel musí dodateční informace získat z jiných externích zdrojů. Tím, že se o *Reactu* nejčastěji mluví jako o knihovně určené především k vytváření uživatelských rozhraní, lze tuto stručnost do určité míry pochopit. Lépe zpracovaná a obsáhlejší dokumentace existuje k *Vue*. V této dokumentaci je zároveň popsán taktéž i vlastní ekosystém. Nejobsáhlejší dokumentaci nicméně poskytuje *Angular*. Tato skutečnost z velké částí vyplývá z toho, že zde se jedná o opravdu komplexní framework, který je potřeba mít podrobněji zdokumentovaný.

Všechny frameworky poskytují dobře zpracovanou referenci na API a navíc pro ně lze nalézt velké množství různorodých příruček, radících, jak daný framework využívat.

U všech frameworku existuje také možnost pro komunitu podílet se na úpravách nebo při tvorbě návodů. Mezi další sledovaná kritéria pro vyhodnocení kvality dokumentace byla skutečnost, zda je v dokumentaci pro framework také *cheatsheet*, který z pohledu některých vývojářů tvoří velmi přínosnou část, protože není nutné procházet vždy celou dokumentaci. Oficiálně byl *cheatsheet* obsažen pouze v dokumentaci pro *Angular*. Navzdory tomu ale nebyly zbylé dva frameworky hodnoceny výrazně hůře, neboť potřebné informace bylo možné v dokumentaci bez větších problémů dohledat.

Jako další oblast pro porovnání byla vybrána nabídka interaktivních příkladů obsažených v samotné dokumentaci. V této oblasti si vedl nejlépe *Vue* a to především díky tomu, že v jeho dokumentací byl k jednotlivým ukázkám přidán i editor, díky kterému si uživatel může danou funkci přímo v dokumentaci vyzkoušet. Ostatní dva testované frameworky tuto možnost nenabízí. Nicméně obsahují odkazy na stránky, na kterých si příklady může uživatel taktéž nasimulovat. Mezí další oblasti bylo zařazeno vyhledávání. Tato oblast byla hodnocena tak, že autor vyhledával předem vybrané formulace a následně porovnával kvalitu získaných odpovědí. Toto nejlépe zvládl *React* a *Vue*, které poskytly přesné a strukturované odpovědi. Třetí testovaný framework, *Angular*, v této disciplíně vykazoval o něco horší výsledky, především z pohledu strukturované odpovědi a zvýraznění dohledaných výsledků. Odpovědi na jednotlivá hledání jsou uvedené v příloze. Rovněž je uveden příklad, kdy docházelo k hledání klíčového slova „*component*“ (Příloha 15, 16, 17).

Posledním sledovaným ukazatelem byla jazyková rozmanitost. V této oblasti kraloval *React*. Tento framework poskytoval v době psaní práce dokumentaci přeloženou do 19 jazyků. Navíc dalších 30 jazykových mutací bylo připravováno. Velký rozdíl byl naopak zaznamenám v případě *Vue*, jehož dokumentace k, toho času, aktuální verzi byla přeložena do dva jazyků. Co se týká *Angularu*, zde existovalo pět jazykových mutací. Z toho důvodu byl hodnocen lepším výsledkem, než kterého dosáhl *Vue*. V poslední řadě je nutné dodat, že ačkoliv se počet jazykových překladů jednotlivých frameworků poměrně značně lišil, všechny tři testované poskytují překlad do jednoho z nejpoužívanějších jazyků – angličtiny. Z tohoto pohledu lze výsledky považovat za příznivé pro zde testované všechny frameworky.

5.1.7 Křivka učení

První oblastí, která byla v tomto směru hodnocena bylo porovnání toho, jak frameworky přistupují k *TypeScriptu*. Předpokladem pro toto hodnocení bylo, že uživatel zná z jazyků pouze HTML, CSS a JS. To znamená, že součástí předpokladu bylo i to, že uživatel se doposud s žádným frameworkem pro práci nesetkal. *TypeScript* lze nazvat jako nový jazyk, který poskytuje nadstavbu Javasriptu a poskytuje mu podporu pro striktní typování. Tuto nadstavbu podporovaly všechny porovnávané frameworky. Rozličnı́ přístup je zde aplikován ze strany *Angularu*, který si ho přímo vynucuje a tím pádem je od jeho uživatelů požadováno ho ovládat. Je sice možné se jeho použití vyhnout, nicméně v praxi platí fakt, že je většinou využíván, a to především díky tomu, že bez něho by bylo použití značně složité, jelikož by nebylo možné využívat deklarace pomocí dekorátorů.

Další hodnocenou disciplínou bylo psaní komponent a logiky těchto komponent. Pod tímto označením si lze představit např. možnosti, jak pracovat se stavem aplikace). Nejsložitější manipulace je v tomto případě s *Angularem*, jelikož, jak již bylo řečeno, vyžaduje tento framework využití *TypeSriptu*. Navíc využívá odlišně specifikované třídy, než je u objektově orientovaných jazyků zvykem. Tím může být práce komplikovanější. Ve *Vue* je možné komponenty tvořit jako objekty obsahující určité vlastnosti, které jsou pro uživatele využitelné. Příchodem *React hooks* v *Reactu* umožňoval komponenty vytvářet čistě funkční. Stále je možné využívat komponenty tříd, ale dnes se již využívají funkční komponenty s *hooky*.

K dalším kritériím hodnocení byla zařazena schopnost vykreslit a psát šablony. K tomuto účelu využívá *React* Javasript, čímž se liší od zbylých dvou hodnocených frameworků, které využívají pro tento účel systém šablon. *React* vykresluje jednotlivé komponenty za pomoci JSX (tzn. Javascript SML). Řeč je o modifikovaném JavaScriptu, jež může připomínat HTML, ale dovoluje použití JavaScriptu. Při hodnocení z pohledu samotného psaní lze psaní ohodnotit jako jednoduché. Nicméně, také díky faktu, že logika komponenty není nijak oddělená od samotného vykreslení se může stávat, že ve složitější komponentně bude vznikat problém s nepřehledností. Zde ale záleží stylu psaní uživatele. Pro psaní lze využívat JSX ale není to nezbytnou podmínku. V případě, kdy se uživatel rozhodne ho nevyužít, bude mít práci o něco těžší. *Angular* i *Vue* k této oblasti přistupují trošku rozdílně a nabízejí jasné oddělení logiky a vykreslení šablony. Psaní šablon

pak probíhá za pomoci HTML, ve kterém je navíc obsažen pouze doménově specifický jazyk. Toto řešení klade na uživatele důraz v tom, aby byl schopen zapamatovat, jak je možné psát jednosměrné vazby z rozhraní ke zdroji. Stejné platí i pro obousměrnou vazbu. Ještě specifičtější je v tomto *Vue*, který kromě těchto šablon dovoluje využít rovněž i JSX.

Jako další oblast pro hodnocení byla zvoleno to, jak dobře lze využívat komponenty. V tomto přístupu nejlépe obstál *React* a to především díky tomu, že využívá klasický import a následně je napřímo využíván JSX. Podobný přístup nabízí také *Vue*. U něho je ale proces o trošku složitější tím, že pro využití komponenty je nutné ji v první řadě přidat do objektu, ve kterém je plánováno ji použít. Ačkoliv je proces trošku složitější, stále lze hovořit o poměrně jednoduchém postupu.

Složitější proces je u *Angularu*. Ten požaduje komponenty registrovat do jednotlivých modelů, bez této funkcionality je nelze využít.

Na základě vyhodnocování dobrá and potaz znalost konceptu vstřikování závislostí (DI). Jedná se složitější techniku, ale můžeme ji označit jako za velice užitečnou. Toto můžeme vidět frameworků které jsou napsané objektově orientovaným jazykem např. *C#* či *PHP*. Podstatou hodnocení bylo zkoumání skutečnosti, zdaje od vývojáře vyžadována znalost tohoto konceptu. Pokud vývojář by rád využil danou techniku v *Angularu*, je zapotřebí ji ovládat a do určité míry jí znát.

Naproti tomu, v případě *Reactu* a *Vue* lze toto řešit za pomoci triviálních konceptů. V rámci uživatelského rozhraní je potřeba ošetřit, že bude korektně probíhat synchronizace. K tomuto slouží vazba mezi daty (data binding). *Angular* i *Vue* dokážou pracovat jak s obousměrnou vazbou, tak také s vazbou jednosměrnou. První jmenovaný typ lze obecně považovat za snadnější pro pochopení, práci i následné spravování. V případně jednosměrného typu je dán předpoklad, kdy jsou vazby neměnné. Z tohoto důvodu je v některých případech třeba vytvořit více kódu. To může být také důvod, kvůli kterému jsou považovány za složitější. V případě *Reactu* jsou přípustné pouze vazby jednosměrné. Čímž v tomto případě odpadá možná složitost, která vychází z toho, že uživatel nemusí být v prvopočátku schopný správně rozlišit, kterou vazbu má použít.

5.1.8 Dostupné knihovny

Nejvíce dostupných knihoven má právě *React*, který uspěl z pohledu velikosti komunity. Jak již víme na množství knihoven má velký vliv komunita. *Angular* a *Vue* je pozadu oproti *Reactu*.

Nicméně i přes menší množství oproti *Reactu* je v obecné rovině knihoven stále mnoho. Větší množství *React* knihoven může být zaměřeno na specifické účely. Dnes již velké množství knihoven mají všechny tři vybrané frameworky.

5.1.9 Ekosystém

Toto kritérium primárně posuzovalo nástroje frameworku, pokud je lze standardně využívat dle potřeby aplikace. Nejméně vyhovujícím se stal *React*. Důvodem je především to, že v jeho případě hovoříme pouze o knihovně. Obsahuje v sobě nástroje pro manipulaci DOM a správu stavu, tím jsou však nástroje v podstatě vyčerpány. Stejné nástroje poskytne také *Vue*, v jeho případě ale ještě navíc poskytuje nástroje pro routování. U obou zmíněných frameworků se uživatel při vyšších nárocích nevyhne využití knihoven třetí strany. Jako nejlepší se ukázal být *Angular*, jež obsahuje veškeré nástroje potřebné k běžnému fungování aplikace.

5.1.10 Flexibilita

Jako nejvíce flexibilní vyšly z porovnání frameworky *React* a *Vue*. V případě *Angularu* se jedná o dosti komplexní framework, z čehož plyne, že je striktnější z pohledu pravidel definovaných pro práci. Oproti prvním dvěma zmíněným nelze *Angular* využít v případech, kdy by do něho chtěl uživatel převést již funkční aplikaci. Zároveň první dva frameworky nabízí více způsobů pro práci v případě tvorby komponent. Zde je u *Angularu* opět možná pouze jedna cesta řešení. Oproti tomu ale *Angular* společně s *Vue* nabízí více řešení pro práci s vazbou dat. Zde na druhou stranu ztrácí *React*, který zde dovoluje pouze jeden způsob.

Co se týká možnosti pro *server-side* vykreslování, zda tuto možnost nabízí všechny tři porovnávané frameworky. Díky tomu jsou všechny vhodné pro vývoj mobilních aplikací.

Nabízí také možnost tzv. multiplatformní vývoj. K tomu ale vyžadují dodatečné nástroje. Z existujících lze jmenovat např. *Ionic*.

5.2 Analýza možnosti využití vybraných frameworků

Díky informacím pramenících z výzkumu během práce byly získány důležité podklady pro vytvoření možností, pro jaké případy lze jednotlivé frameworky využít. Na to, pro co se jaký framework hodí, lze nahlížet z několika směrů. Mezi nejčastější směr patří hodnocení, do jaké míry se framework hodí pro využití v aplikaci z pohledu jejích potřeb během jejího života. Vzhledem k vývoji aplikace je kladen důraz na to, aby framework nabízel dostatečné nástroje a funkce.

Můžeme říct, že všechny frameworky vyhovují potřebám webových aplikací. Hlavně díky množství nástrojů, které frameworky nabízí. Z tohoto důvodu bylo k analýze zahrnuto hodnocení samotných vývojářů vycházející nejen s každodenní zkušeností a potřeb, ale i preferencí jednotlivých účastníků.

Práce obsahuje analýzu možností využití ve zmiňovaných pohledech první podkapitola obsahuje zohlednění potřeb aplikace, druhá obsahuje možnosti z pohledu vývojáře (resp. vývojářů) a třetí obsahuje doporučení pro business a webové aplikace.

5.2.1 Z pohledu potřeb aplikace

Všechny testované frameworky byly pro test zamýšlené především pro jednostránkovou webovou aplikaci. Z tohoto pohledu se pro práci hodí všechny z testovaných. Každý z frameworků je schopný nabídnout také SEO optimalizaci na velmi kvalitní úrovni pro případ, kdy by byly využité pro složitější implementace, mezi které lze zmínit např. e-shop. Lze je ale rovněž využít i pro případ, kdy se uživatel rozhodne pro vícestránkovou aplikaci. Pro tento účel poskytnou důležitou funkčnost ve formě vykreslování dat na serveru.

Další oblastí, ve které najdou frameworky využití může být například i vývoj aplikací pro mobilní zařízení. Aby bylo možné v této oblasti vyhodnotit použitelnost, využil autor práce existující projekty *RealWorld* a *TodoMVC*. Tyto projekty pochází z řad komunity. Vybrány byly především proto, protože se jejich technika měření z velké části podobá

technice použité v této práci. Co se týká *RealWorld*, zde se jedná o větší počet aplikací, které se soustředí na zpodobnění portálu medium.com. Aplikace jsou psané za pomoci rozličných frameworků. Projekt *TodoMVC* se pak zaměřuje na aplikaci a správu seznamu úkolů. Z výsledků pomocí *Lighthouse* lze pozorovat, že výkon jako takový není přímo závislý na konkrétním použitém frameworku, ale z velké části závisí na schopnosti vývojáře danou aplikaci optimalizovat. Právě díky této skutečnosti není zcela možné jednoznačně určit, který framework je v tomto směru lepší.

Hodnocené frameworky najdou využití také v případě, kdy je uživatel zaměřen na nativní mobilní aplikace. V tomto případě je ale nutné použít dodatečné nástroje. V případě *Reactu* lze hovořit o *React Native*. Pro *Angular* je k dispozici *Native* Script. Pro *Vue* se pak používá nástroj *Vue Native*. Díky tomu, že je *NativeScript* cílen na to poskytnout podporu pro větší množství frameworků pro vývoj na více platforem, lze ho využít také ve *Vue*. Pro *React* lze využít *Ionic*.

Pokud je potřeba z důvodu modernizace zavést framework do existující aplikace je nejlepší využívat *React* anebo *Vue*. Tímto postupem lze přepsat již existující aplikaci do některého z vybraných frameworků. Touto cestou se lze vydat v kterémkoliv jazyce nebo za pomoci kteréhokoliv frameworku. Nicméně zde platí omezení, že se musí jednat o webovou aplikaci. Z pohledu jednoduchosti řešení vychází nejlépe *Vue*. Pro integraci tohoto frameworku je potřeba jen vložit CDN do HTML. Obdobně lze postupovat i v případě *Reactu*. Drobný rozdíl zde je, že pokud by uživatel chtěl použít také JSX, nevyhne se potřebě provést lehčí konfiguraci. U *Angularu* se mluví o komplexním frameworku. Z tohoto důvodu také tuto možnost nedává k dispozici.

5.2.2 Z pohledu vývojáře

Preference každého z uživatelů pramení zejména z jeho doposud nabytých zkušeností. Právě z tohoto důvodu se tento stav pro každého liší. Lze zmínit vývojáře, kteří disponují znalostmi základních jazyků jako jsou např. HTML, CSS nebo JS. Lze ale rovněž mluvit o vývojáři, který pracoval s několika frameworky a ovládá více jazyků. Proto je pro další vyhodnocení potřeba toto zahrnout do vstupů.

Nejlepšího výsledku v oblasti křivky učení dosáhl *Vue*, jenž se v tomto případě ukázal být nejjednodušší. Tento framework se zakládá na využívání základních jazyků

HTML, CSS a JS. Dokumentace tohoto frameworků navíc obsahuje mimo jiné i interaktivní ukázky, což je z pohledu začínajícího vývojáře velmi užitečné pro pochopení podstaty fungování kódu. Pokud se jedná o začínající vývojáře je možné považovat *Vue* jako vhodný framework. Zároveň i *React* je vhodný pro začínající vývojáře.

Vychází z Javascriptu, a to v případě kdy je potřeba využít JSX. A to vyšší nároky na pochopení uživatele. Mezi matoucí může být i zpráva stavu aplikace. Na druhém konci hodnocení se umístil *Angular*. Díky své komplexnosti bývá složitý pro začínající vývojáře.

Zde opět záleží na konkrétním vývojáři. Pokud má již nějaké zkušenosti z jiných frameworků či pracoval s objektově orientovanými jazyky, může to pro něj znamenat výhodu a v konečné podstatě mu tento přístup může vyhovovat. Nicméně i v případě, kdy by se jednalo o začínajícího programátora, není vždy podmínku, že bude vyhledávat jednodušší framework. Doporučení by se tedy mělo odvíjet zároveň od preferencí daného uživatele. Ne vždy se přitom musí jednat přímo o preference jednotlivého programátora. Obvykle jde o názor skupiny uživatelů, kteří pracují v týmu.

V *Reactu* je velice snadné vytváření komponent jak funkčních, tak class komponent. Další výhodou může být využívání *TypeScriptu* nebo využívání šablon. Dalším bodem, který stojí za zmínku je, že *React* je vhodný v případě, kdy se uživatel chce spolehnout na aktivitu a velikost komunity. *React* není framework ale knihovna. Často využívá knihovny třetích stran. Pokud vývojáři směřují na jednoduchou aplikaci, je lepší zvolit framework *Vue*. *Vue* framework je velice jednoduchý a přehledný. Jazyky jsou v něm jasně oddělené. Vývojáři mohou vytvářet komponenty jedno souborovou cestou, kde jsou bloky HTML, CSS a JS. Je využívat také *TypeScript*. Pokud vývojáře chtějí využívat stabilní vývoj a nechtějí využívat knihovny třetích stran, je dobré zvolit *Angular* který má striktně daná pravidla. U *Angularu* se komponenty rozdělují do souborů. Stylování se využívá jeden soubor pomocí preprocesoru SCSS. Při společná vlastnost s frameworkem *Vue* je využívání šablony za pomoci HTML. Výhodou udržitelnosti a čitelnosti kódu může být využívání striktních pravidel. Tvořené aplikace těmito frameworky jsou defacto identické.

5.2.3 Z pohledu doporučení pro business a webové aplikace

K testování frameworků vícestránkové aplikace tak i jednostránkové aplikace lze považovat všechny vybrané frameworky. Pro implementaci e-shopu mají frameworky

velmi dobrou SEO optimalizaci. U těchto implementací se klade vysoký nárok na indexaci pro vyhledávací nástroje. Další funkcionalitou MPA (Multi Page Aplikace) je vykreslování na straně serveru.

Nicméně v případech, kdy je uvažováno zavedení frameworků do již existující aplikace z důvodu modernizace, refaktoringu, či pro účely využití konkrétních funkcionalit daného frameworků, lze z jmenovaných využít *React* nebo *Vue*. U *Angularu* se jedná o komplexní framework, z toho důvodu tyto možnosti nenabízí. *React* je oproti tomu knihovna, která často využívá knihovny třetích stran. Hlavní funkcionalitou je například využívání formulářů. Díky této skutečnosti lze aplikaci přepisovat do *Reactu* nikoliv nutně celou ale pouze po částech.

Stejnou cestou lze postupovat v případě, kdy si uživatel zvolí *Vue*. I ten je, co se týká jednoduchosti použití, vhodnější než *Angular*. Popsaný způsob je aplikovatelný defacto všechny projekty, bez ohledu na jazyk nebo framework, ve kterém jsou napsány. Jediné omezení, kdy tento postup nelze aplikovat je v případě business nebo webových aplikací. Pro integraci *Vue* je potřeba pouze vložit CDN do HTML. V podstatě totožná situace je i v případě využití *Reactu*. Zde je ale drobná odlišnost v tom, že v případě potřeby využívat JSX je potřeba konfigurace.

Tyto Frameworky se většinou používají při složitějších aplikacích, to znamená, že jsou rozdělené na jak na frontendovou tak i backendovou část, kdy se data odesílají a dotazují na serverovou část. Jak pro webové, tak i business aplikace lze všechny jmenované frameworky využívat s tím, že v konečném důsledku z velké části záleží na konkrétním určení dané aplikace a rovněž také na preferencích vývojáře.

6 Závěr

Teoretická část práce se zabývá problematikou vývoje webových aplikací. Následně jsou popsány technologie, a jejich náležitosti spojené s vývojem webových aplikací pomocí jazyka JavaScript a JavaScript frameworků.

V praktické části práce byli vybraný tři nejpoužívanější frameworky ke zhodnocení, a to na základě dotazníku *State of JS 2021*. Výběr se zakládal na dostupných informacích o jednotlivých frameworcích, včetně názorů uživatelů z řad vývojářů získaných v rámci provedeného výzkumu *State of JS 2021* a *Developer Survey 2022*. Dalším krokem bylo určení kritérií pro hodnocení zvolených frameworků, které se zaměřilo na aspekt výkonu, komunity, použitelnosti a vývoje.

Následným krokem bylo hodnocení pomocí metody vícekriteriální analýzy variant, kde byly určené váhy kritérií Saatyho metodou. Ohodnocení kritérií vycházelo od Senior vývojářů. Z výsledků lze usuzovat, že *React* poskytuje celkový užitek nejvyšší, především díky tomu, že má oproti zmíněným konkurentům největší podporu komunity. *Vue* se zde umístil na druhém místě, zejména díky malé komunitě. Zároveň byl však vyhodnocen jako jednoduchý pro nově začínající vývojáře. Poslední třetí místo obsadil *Angular*, který má vysokou křivku učení a nedisponuje tak rozsáhlou základnou ze strany komunity jako *React*.

Poslední hodnocení bylo zaměřeno na otázku doporučení těchto frameworků k využití pro webové a business aplikace. Jak vyplynulo z předchozího výzkumu, všechny frameworky byly vyhodnoceny jako vhodné jak pro SPA (Single Page aplikace), tak pro MPA (Multi Page aplikace). Hlavním kritériem pro výběr daného frameworku je otázka, jaká aplikace má pomocí něj vzniknout. Záleží především na tom, jak má aplikace má být složitá a také dle preferencí vývojáře. *React* a *Vue* jsou dle křivky učení jednodušší. Oproti tomu *Angular* je komplexnější, se striktně danými pravidly.

Další porovnání může být ovlivněno rozvojem těchto frameworků a jejich novými funkcionalitami, které lze zjistit dle budoucího hodnocení na průzkumech v *State of JS 2022* a *Developer Survey 2023*.

7 Seznam použitých zdrojů

[1] SCOTT, Emmit A. SPA design and architecture: understanding single-page web applications. Shelter Island, NY: Manning, 2016. ISBN 978-1617292439.

[2]ASP.NET-Single-PageApplications:BuildModern,ResponsiveWebAppswith ASP.NET [online]. [cit. 2022-04-30]. Dostupné z: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx.

[2] BROWN, Tiffany B., Kerry BUTTERS a Sandeep PANDA. HTML5 okamžitě: [ovládněte HTML5 za víkend]. Brno: Computer Press, 2014. ISBN 9788025142967

[3] HTML 5.2 [online]. [cit. 2022-05-02]. Dostupné z: https://www.w3.org/TR/2017/REC-html52-20171214/

[4] GASSTON, Peter. CSS3. Přeložil Ondřej BAŠE. Brno: Computer Press, 2016. ISBN 9788025146415.

[5] CSS preprocessor [online]. [cit. 2022-05-02]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/CSS_preprocessor

[6] Javascript [online]. [cit. 2022-05-02]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[7] The10most popular programming languages, according to the 'Facebook for programmers' [online]. [cit. 2022-05-02]. Dostupné z: https://www.businessinsider.com/the-10-most-popular-programming-languages- according-to-github-2018-10#1-javascript-10.

[8] ZAKAS, Nicholas C. JavaScript pro webové vývojáře. Brno: Computer Press, 2009. Programujeme profesionálně. ISBN 978-80-251-2509-0.

[9] MARGORÍN, Marián. JQuery bez předchozích znalostí. Brno: Computer Press, 2011. ISBN 978-80-251-3379-8.

[10] ITnetwork: Lekce 1 - Úvod do knihoven a frameworků [online]. David Čápka, 2014 [cit. 2022-12-03]. Dostupné z: https://www.itnetwork.cz/php/knihovny/php-tutorial-uvod-do- knihoven-a-frameworku .

[11] jQuery [online]. [cit. 2022-11-15]. Dostupné z: https://jquery.com/

[12] Architecture overview [online]. [cit. 2022-04-30]. Dostupné z: https://angular.io/guide/architecture

[13] JANSEN Remo. Learning TypeScript. Birmingham, Spojené království: Packt Publishing, 2015. ISBN 1783985550.

[14] FAIN, Yakova Anton MOISEEV.Angular2 development with TypeScript.Shel- ter Island, NY: Manning Publications Co., 2017. ISBN 9781617293122.

[15] Introduction to modules [online]. [cit. 2022-04-30]. Dostupné z: https://angular.io/guide/architecture-modules

[16] FREEMAN, Adam. Pro Angular 6. New York, NY: Springer Science Business Media, 2018. ISBN 978-1484236482.

[17] Introduction to components [online]. [cit. 2022-04-30]. Dostupné z: https://angular.io/guide/architecture-components

[18] Lifecycle sequence [online]. [cit. 2022-05-02]. Dostupné z: https://angular.io/guide/lifecycle-hooks

[19] Aquick intro to Dependency Injection: whatitis, and when to use it [online].[cit. 2022-04-30]. Dostupné z: https://medium.freecodecamp.org/a-quick-intro-to- dependency-injection-what-it-is-and-when-to-use-it-7578c84fa88f

[20] Routing & Navigation [online]. [cit. 2022-04-30]. Dostupné z: https://angular.io/guide/router

[21] Binding syntax: An overview [online]. [cit. 2022-04-30]. Dostupné z: https://angular.io/guide/template-syntax

[22] Angular NgFor: Everything you need to know [online]. [cit. 2022-05-02]. Do- stupné z: https://malcoded.com/posts/angular-ngfor/

[23] DatePipe [online]. [cit. 2022-05-02]. Dostupné z: https://angular.io/api/common/DatePipe

[24] Angular 4 - Pipes [online]. [cit. 2022-05-02]. Dostupné z: https://www.tutorialspoint.com/angular4/angular4_pipes.htm

[25] Hello World, <angular/> is here [online]. [cit. 2022-05-02]. Dostupné z: http://misko.hevery.com/2009/09/28/hello-world-angular-is-here/

[26] A sneak peek at the radically new Angular 2.0 [online]. [cit. 2022-05-02]. Do- stupné z: https://jaxenter.com/angular-2-0-112094.html

[27] Angular 2.0 announcement backfires [online]. [cit. 2022-05-02]. Dostupné z: https://jaxenter.com/angular-2-0-announcement-backfires-112127.html

[28] Angular - One framework. Mobile & desktop. [online]. [cit. 2022-05-02]. Do- stupné z: https://github.com/angular/angular

[29] WIERUCH, Robin. The Road to learn React: Your journey to master plain yet pragmatic React.js. CreateSpace Independent Publishing Platform, 2018. ISBN 978-1986338820.

[30] Tutorial: Intro to React [online]. [cit. 2022-05-02]. Dostupné z: https://reactjs.org/tutorial/tutorial.html#what-is-react

[31] Components and Props [online]. [cit. 2022-05-02]. Dostupné z: https://reactjs.org/docs/components-and-props.html

[32] React.Component [online]. [cit. 2022-05-02]. Dostupné z: https://reactjs.org/docs/react-component.html

[33] React Lifecycle Methods – A Deep Dive [online]. [cit. 2022-05-02]. Dostupné z: https://programmingwithmosh.com/javascript/react-lifecycle-methods/

[34] Draft: JSX Specification [online]. [cit. 2022-05-02]. Dostupné z: https://facebook.github.io/jsx/

[35] Fedosejev, Artemij. React.js essentials: a fast-paced guide to designing and buil- ding scalable and maintainable web apps with React.js. First published. Bir- mingham: Packt Publishing, 2015. x, 183 stran. Open source community expe- rience distilled. ISBN 978-1-78355-162-0.

[36] A Beginners Guide to Redux [online]. [cit. 2022-05-02]. Dostupné z: https://www.gistia.com/beginners-guide-redux/

[37] The History of React.js on a Timeline [online]. [cit. 2022-05-02]. Dostupné z: https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/

[38] React: A declarative, efficient, and flexible JavaScript library for building user interfaces [online]. [cit. 2022-05-02]. Dostupné z: https://github.com/facebook/react

[39] CHAU, Guillaume. Vue.js 2 Web Development Projects: Learn Vue.js by buil- ding 6 web apps. Packt Publishing, 2017. ISBN 978-1787127463.

[40] Components Basics [online]. [cit. 2022-05-09]. Dostupné z: https://vuejs.org/v2/guide/components.html

[41] Understanding Vue.js Lifecycle Hooks [online]. [cit. 2022-05-09]. Dostupné z: https://alligator.io/vuejs/component-lifecycle/

[42] Single-page App vs. Multi-page App [online]. [cit. 2022-09-17]. Dostupné z: https://lvivity.com/single-page-app-vs-multi-page-app

[43] The architecture of Angular [online]. [cit. 2022-09-17]. Dostupné z: https://ducmanhphan.github.io/2019-02-23-The-architecture-of-Angular/

[44] Node.JS v Shoptet Premium [online]. [cit. 2022-09-19]. Dostupné z: https://kafemlejnek.tv/dil-57-node-js-v-shoptet-premium/

[45] AppRoutingModule [online]. [cit. 2022-09-19]. Dostupné z: https://angular.io/guide/router

[46] React.Component [online]. [cit. 2022-09-19]. Dostupné z: https://reactjs.org/docs/react-component.html

[47] GETTING STARTED WITH REDUX [online]. [cit. 2022-09-19]. Dostupné z: https://www.indiehackers.com/post/redux-and-drizzle-day-75-7eae7c5d32

[48] What is Vuex [online]. [cit. 2022-09-19]. Dostupné z: https://vuex.vuejs.org.

[49] Builtwith [online]. [cit. 2022-09-20]. Dostupné z: https://builtwith.com/

[50] ROUSE, Margaret. 2016. Single-page application (SPA). TechTarget [online]. 2016 [cit. 2021-09-22]. Dostupné z: https://whatis.techtarget.com/definition/single-page-application- SPA

[51] OPENJS FOUNDATION. 2011. What is npm? Node.js [online]. 2011 [cit. 2021-09-22]. Dostupné z: https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/.

[52] TUTORIALS POINT. 2020. Node.js - Introduction. Tutorials Point [online]. 2020 [cit.2022-09-22].Dostupné:
https://www.tutorialspoint.com/nodejs/nodejs_introduction.html.

[53] MÁCA, Jindřich. 2018. Úvod do Node.js. ITNetwork [online]. 2018 [cit. 2022-07-25]. Dostupné z: https://www.itnetwork.cz/javascript/nodejs/uvod-do-nodejs

[54] npm trends. 2022. npm trends. [Online] 2022. https://www.npmtrends.com/react-vs-@angular/core-vs-vue-vs-ember-source-vs-svelte-vs-preact-vs-@stimulus/core-vs-alpinejs-vs-@polymer/lit-element.

[55] State of Js. 2021 Front-end frameworks and libraries
2022. [Online] 2022. https://2021.stateofjs.com/en-US/libraries/front-end-frameworks.

[56] Stack Overflow . 2021. Programming, Scripting, and Markup Languages. Stack Overflow Developer Survey 2022. [Online] 2022. https://survey.stackoverflow.co/2022

[57] Stack Overflow. 2022. Developer Survey 2022. Stack Overflow Developer Survey 2022. [Online] 2022. https:// https://survey.stackoverflow.co/2022/.

[58]. 2022. Demographics. Stack Overflow Developer Survey 2022. [Online] 2022. https://survey.stackoverflow.co/2022/#developer-profile-demographics.

[59] 2022. Job Factors. Stack Overflow Developer Survey 2022. [Online] 2022. https://survey.stackoverflow.co/2022/#work.

[60]. 2022. Visiting Stack Overflow. Stack Overflow Developer Survey 2022. [Online] 2022. https://survey.stackoverflow.co/2022/#community.

[61]Greif, Sacha a Benitte, Raphaël. 2021. Demographics. State of JS 2021. [Online] 2022. https://2021.stateofjs.com/en-US/demographics.

[62]. 2021. Front-end Frameworks. State of JS 2021. [Online] 2022. https://2021.stateofjs.com/en-US/libraries/front-end-frameworks.

[63]. 2021. State of JS 2021. State of JS 2021. [Online] 2022. https://2021.stateofjs.com/en-US/.

[64]. Casteleyn, Sven, Dolog, Peter a Pautasso, Cesare. 2016. Current Trends in Web Engineering. místo neznámé : Springer, 2016. 3319469630.

[65] Subreddit Stats. 2022. Subreddit Stats. [Online] 2022. https://subredditstats.com.

[66] Bc. David Pocar . 2020. Vývoj webových aplikací pomocí JavaScript frameworků 2022.

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam kódu

8.4 Seznam grafů

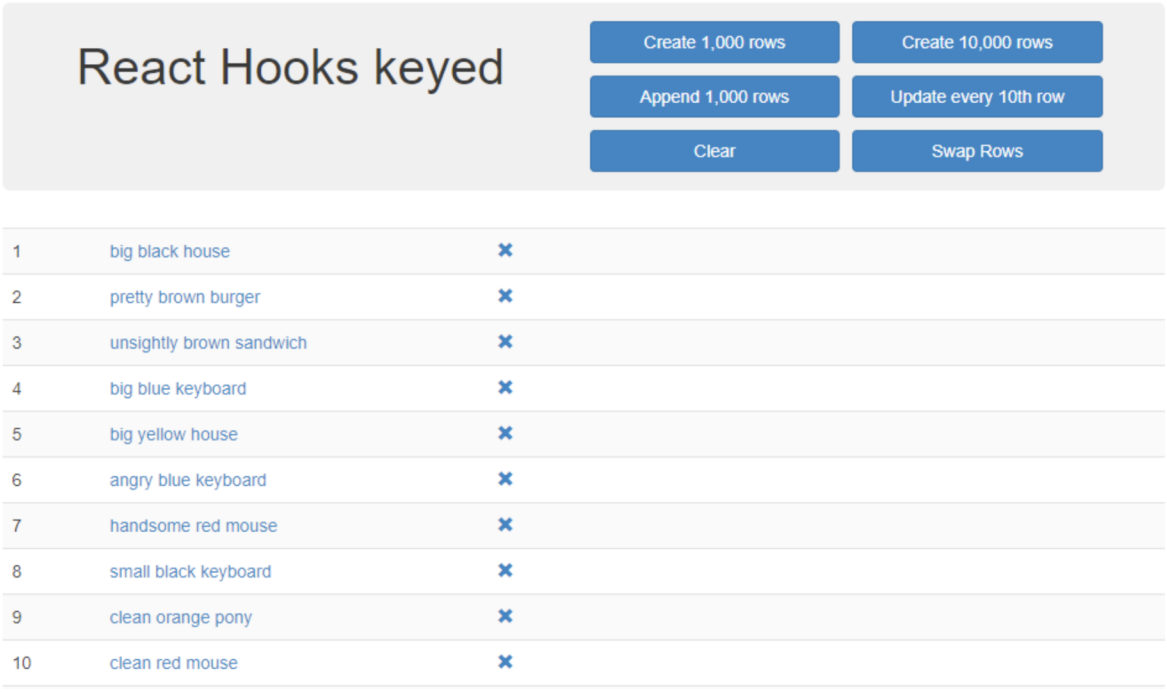
8.5 Seznam vzorců

8.6 Seznam použitých zkratek

API	Aplication Programming Interface - Aplikační programovací rozhraní
CLI	Command Line Interface - Rozhraní příkazového řádku
AJAX	Asynchronous JavaScript and XML
NPM	Node.js package manager
CSS	Cascading Style Sheets - Kaskádové styly
DOM	Document Object Model - Objektový model
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protokol
JS	JavaScript

JSON	JavaScript Object Notation
JSX	JavaScript Extension - Rozšíření JavaScript
MB	Megabyte
MVC	Model-View-Controller
MVVM	Model-View-VieModel
REST	Representational State Transfer
SASS	Syntactically Awesome Style Sheets - Syntaktické styly
SPA	Single Page Application - Jednostránková aplikace
MPA	Multi Page Applicaton – Vícestránková aplikace
XML	eXtensible Markup Language

Přílohy



Příloha A - Ukázka aplikace projektu v js-framework-benchmark

Name Duration for...	vue-v3.2.26	angular-v13.0.0	react-v17.0.2
Implementation notes			
create rows creating 1,000 rows	105.2 ±3.5 (1.00)	115.6 ±2.1 (1.10)	118.0 ±2.7 (1.12)
replace all rows updating all 1,000 rows (5 warmup runs).	90.8 ±0.8 (1.00)	101.7 ±0.8 (1.12)	106.7 ±1.1 (1.17)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16x CPU slowdown.	184.1 ±4.6 (1.11)	166.5 ±2.8 (1.00)	207.7 ±3.0 (1.25)
select row highlighting a selected row. (no warmup runs). 16x CPU slowdown.	33.2 ±1.1 (1.00)	58.7 ±3.2 (1.77)	94.1 ±2.1 (2.83)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	48.9 ±0.5 (1.00)	333.3 ±1.9 (6.82)	329.1 ±1.4 (6.73)
remove row removing one row. (5 warmup runs).	21.5 ±0.2 (1.07)	20.0 ±0.5 (1.00)	22.3 ±0.5 (1.11)
create many rows creating 10,000 rows	1,008.6 ± 14.3 (1.00)	1,108.8 ± 18.7 (1.10)	1,386.7 ± 26.6 (1.37)
append rows to table appending 1,000 to a table of 1,000 rows. 2x CPU slowdown.	199.2 ±4.1 (1.00)	232.2 ±2.2 (1.17)	245.5 ±7.7 (1.23)
clear rows clearing a table with 1,000 rows. 8x CPU slowdown.	66.8 ±1.2 (1.00)	144.9 ±3.9 (2.17)	75.3 ±0.8 (1.13)
geometric mean of all factors in the table	1.02	1.51	1.59

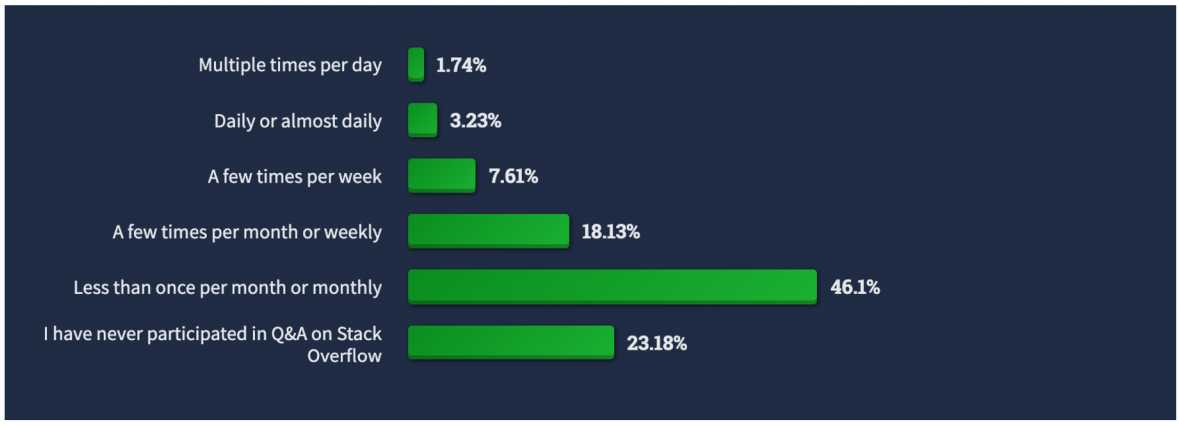
Příloha B - Výsledky měření výkonu projektu v js-framework-benchmark

Name	vue-v3.2.26	angular-v13.0.0	react-v17.0.2
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,105.4 ± 0.0 (1.00)	2,820.4 ± 2.5 (1.34)	2,576.1 ± 9.4 (1.22)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	195.3 ±0.0 (1.00)	294.5 ±0.0 (1.51)	274.4 ±0.0 (1.40)
geometric mean of all factors in the table	1.00	1.42	1.31

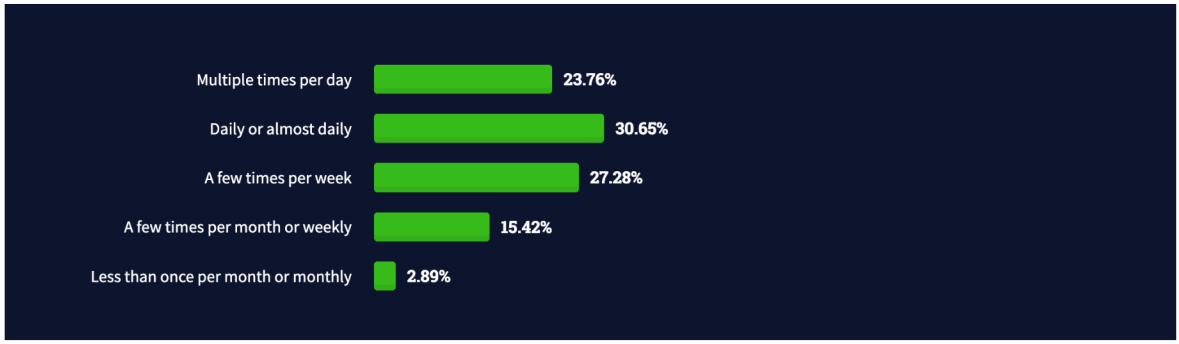
Příloha C - Výsledky měření výkonu projektu v js-framework-benchmark

Name	vue-v3.2.26	angular-v13.0.0	react-v17.0.2
ready memory Memory usage after page load.	1.4 (1.00)	1.7 (1.26)	1.5 (1.05)
run memory Memory usage after adding 1000 rows.	3.2 (1.00)	3.9 (1.23)	4.2 (1.32)
update every 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	3.2 (1.00)	4.0 (1.25)	4.8 (1.49)
replace 1k rows (5 cycles) Memory usage after clicking create 1000 rows 5 times	3.2 (1.00)	4.4 (1.37)	4.5 (1.42)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	1.4 (1.00)	2.4 (1.68)	2.1 (1.47)
geometric mean of all factors in the table	1.00	1.35	1.34

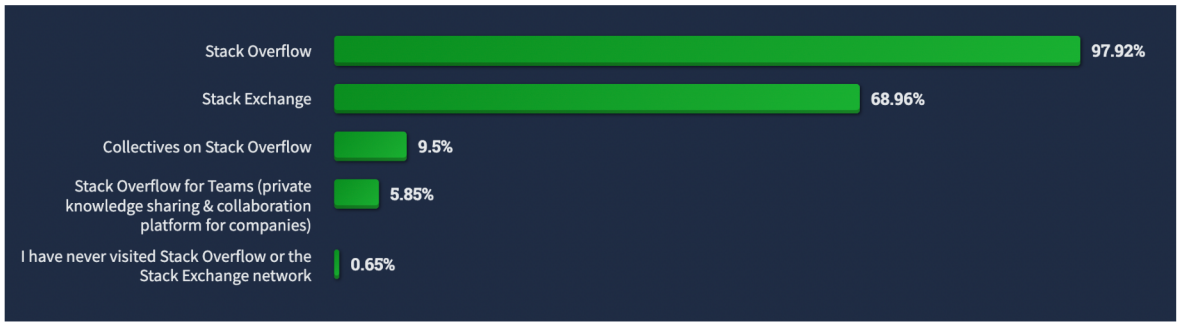
Příloha D - Výsledky měření výkonu projektu v js-framework-benchmark



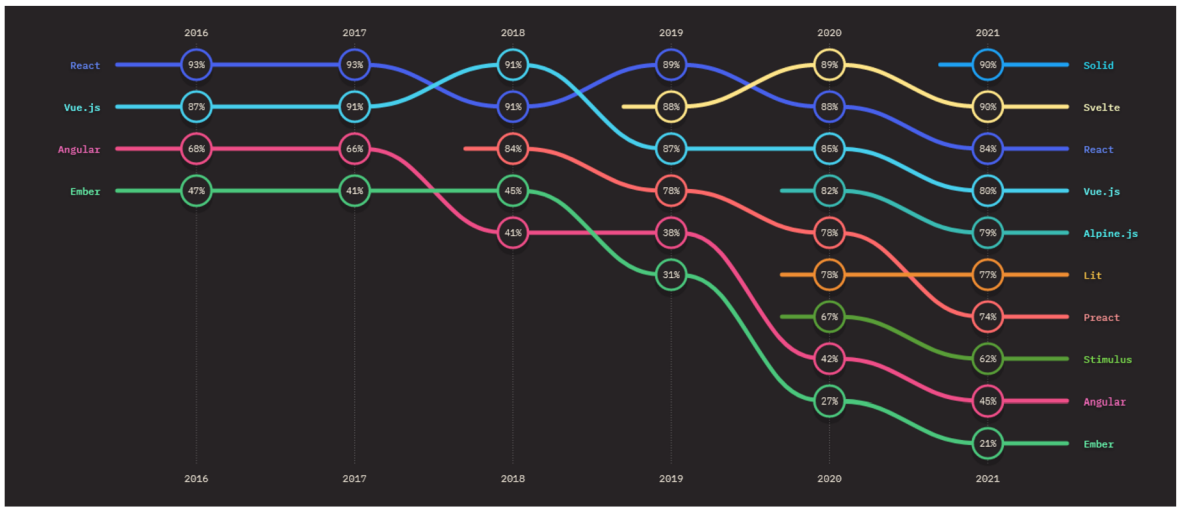
Příloha E – Frekvence účasti na webu Stack Overflow [56]



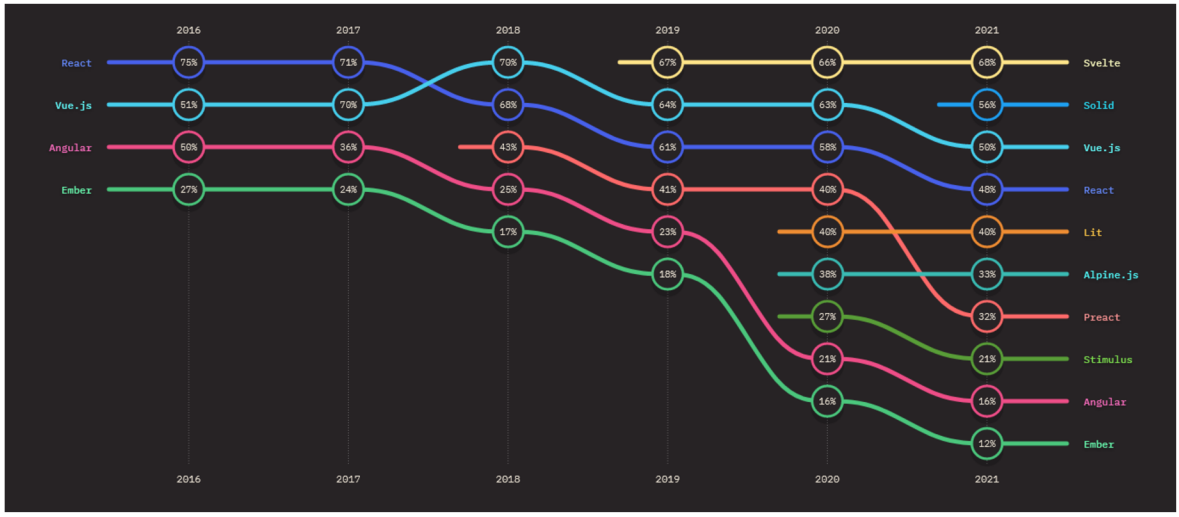
Příloha F – Frekvence návštěvnosti webu Stack Overflow [57]



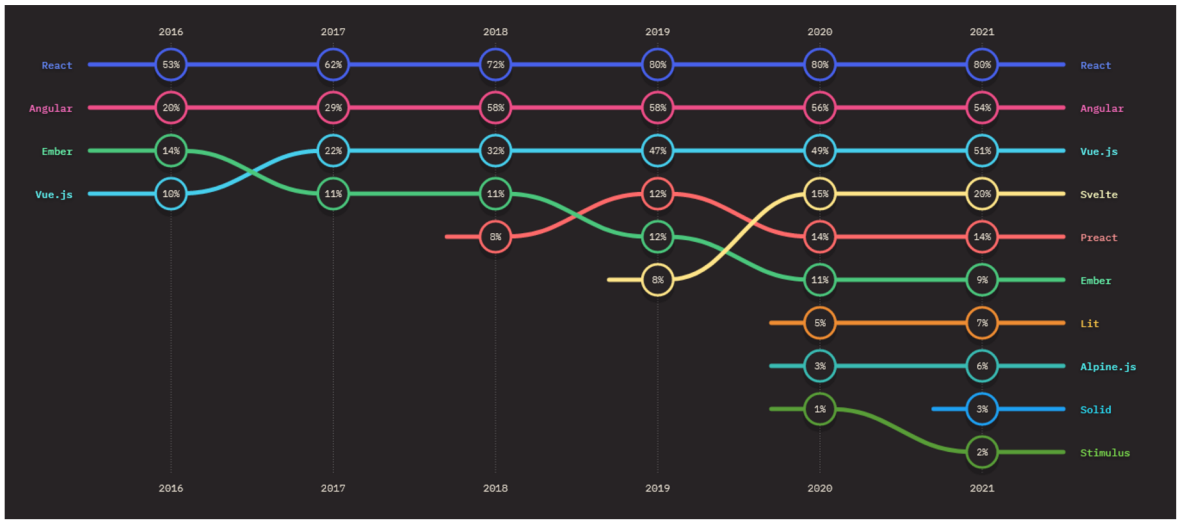
Příloha G – Návštevnost webu Stack Overflow a Stack Exchange [58]



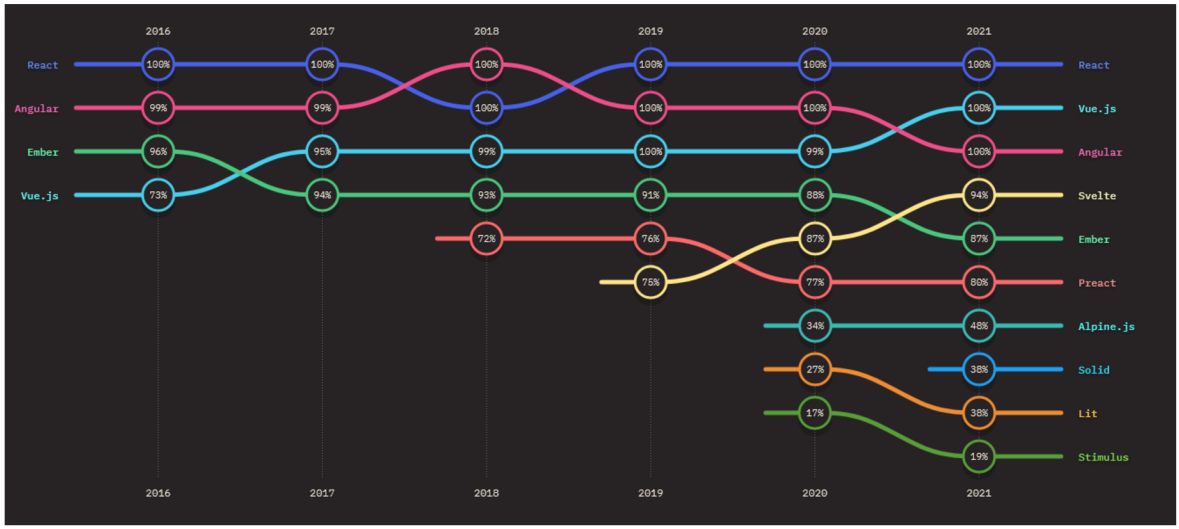
Příloha H - Míra spokojenosti vývojářů s JS frameworky [61]



Příloha I - Míra zájmu vývojářů s JS frameworky [61]



Příloha J - Míra užití JS frameworků vývojáři [61]



Příloha K - Míra známosti JS frameworků vývojáři [61]

r/reactjs stats

Title: /r/ReactJS - The Front Page of React

Description: A community for learning and developing web applications using React by Facebook.

Subscribers	Rank	Comments Per Day	Rank	Posts Per Day	Rank
322 520	2064	143	3855	48	4515

Comments Per Subscriber	Rank	Gildings Per Subscriber	Rank
0.000018	25271	0.000050	10495

Posts Per Subscriber	Rank	Post Votes	Rank	Comments	Rank
0.000006	10370	132 297	6885	49 547	2526

Post Gildings	Rank
16	2424

Příloha L - Statistiky subredditu r/reactjs [65]

r/vuejs stats

Title: Vue.js - The progressive Javascript framework

Description: Vue.js is a library for building interactive web interfaces. It provides data-reactive components with a simple and flexible API.

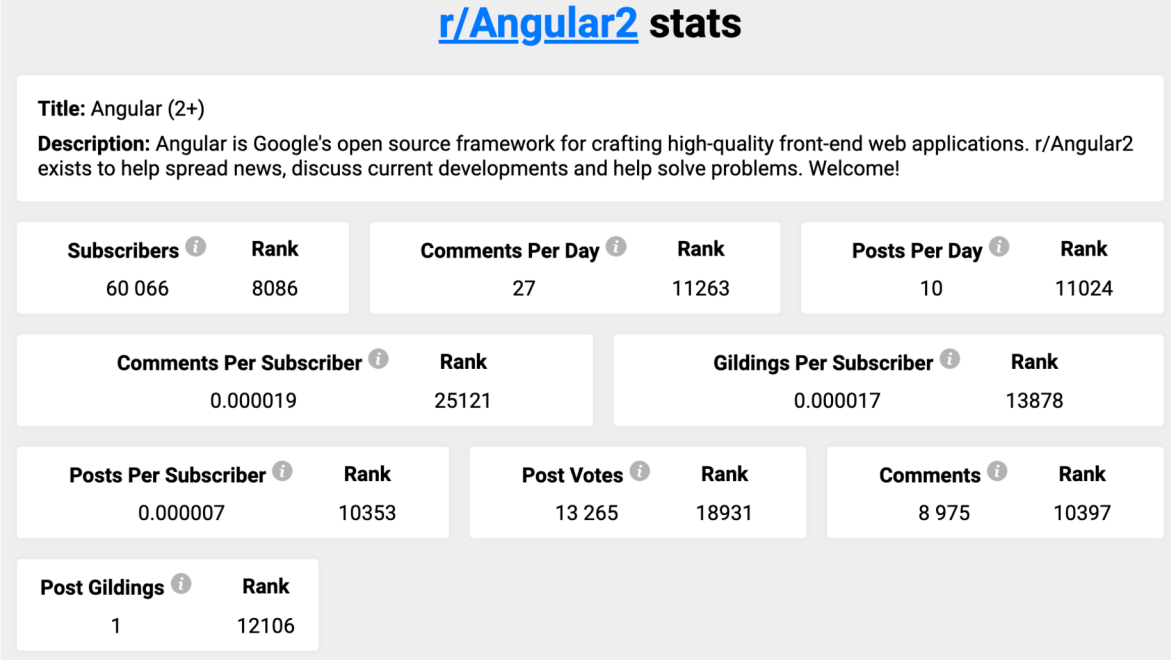
Subscribers	Rank	Comments Per Day	Rank	Posts Per Day	Rank
88 190	6164	55	7510	16	8038

Comments Per Subscriber	Rank	Gildings Per Subscriber	Rank
0.000026	21776	0.000057	9950

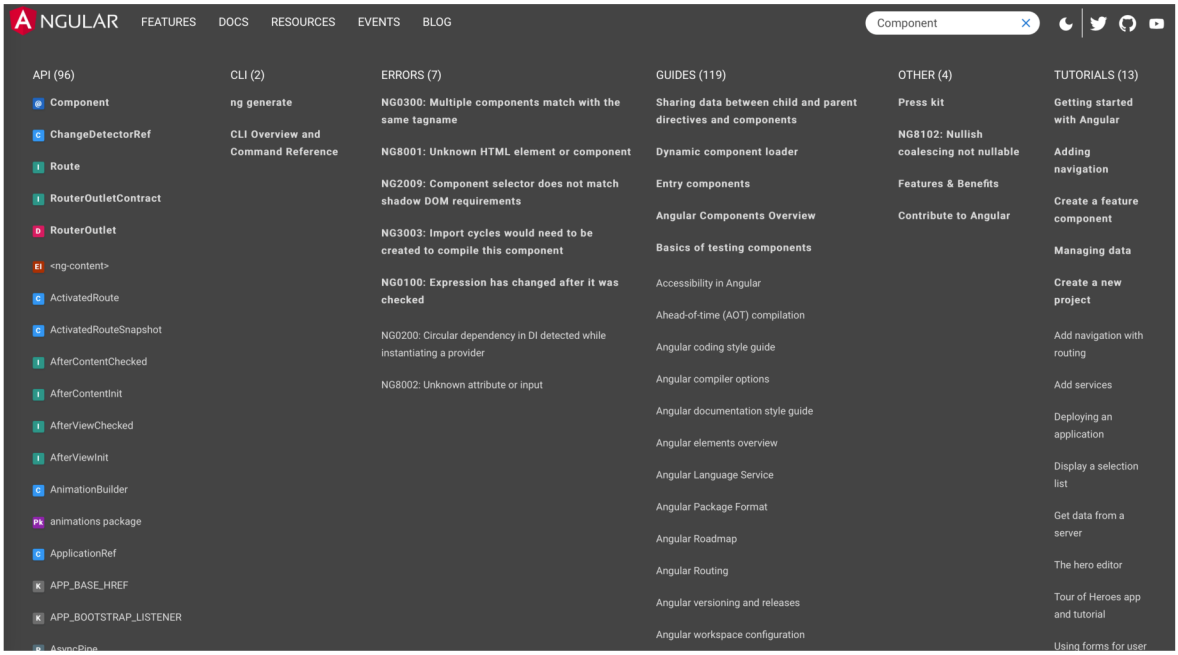
Posts Per Subscriber	Rank	Post Votes	Rank	Comments	Rank
0.000008	4144	28 305	14815	13 609	7827

Post Gildings	Rank
5	5813

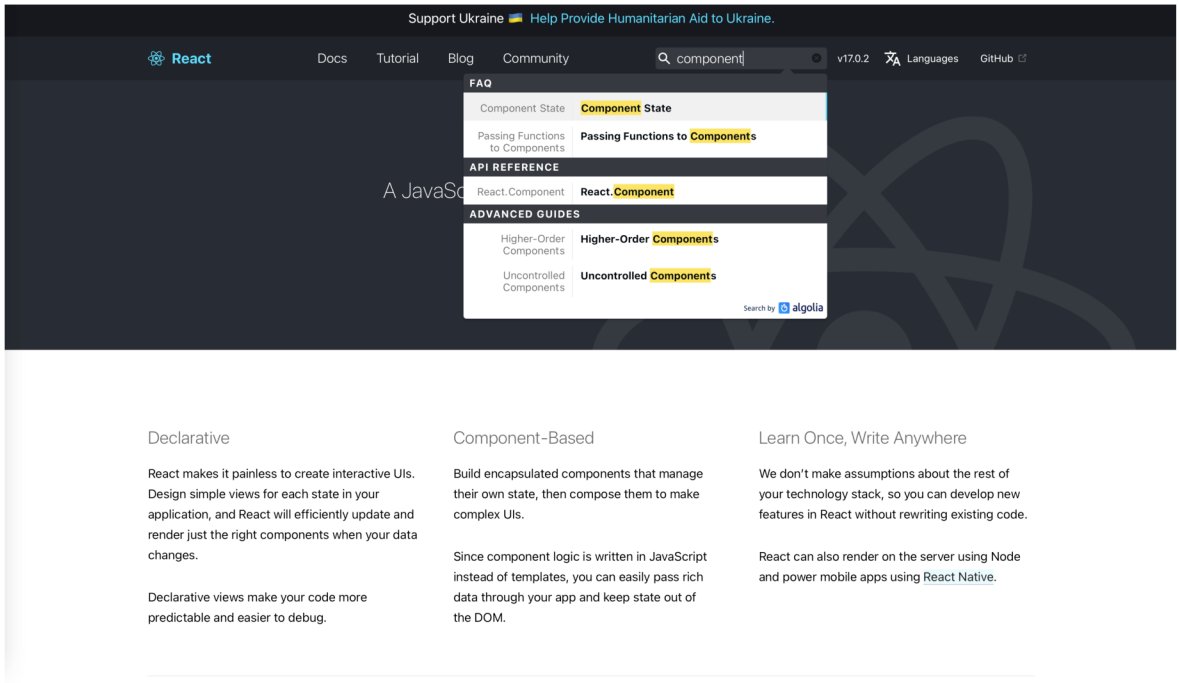
Příloha M - Statistiky subredditu r/vuejs [65]



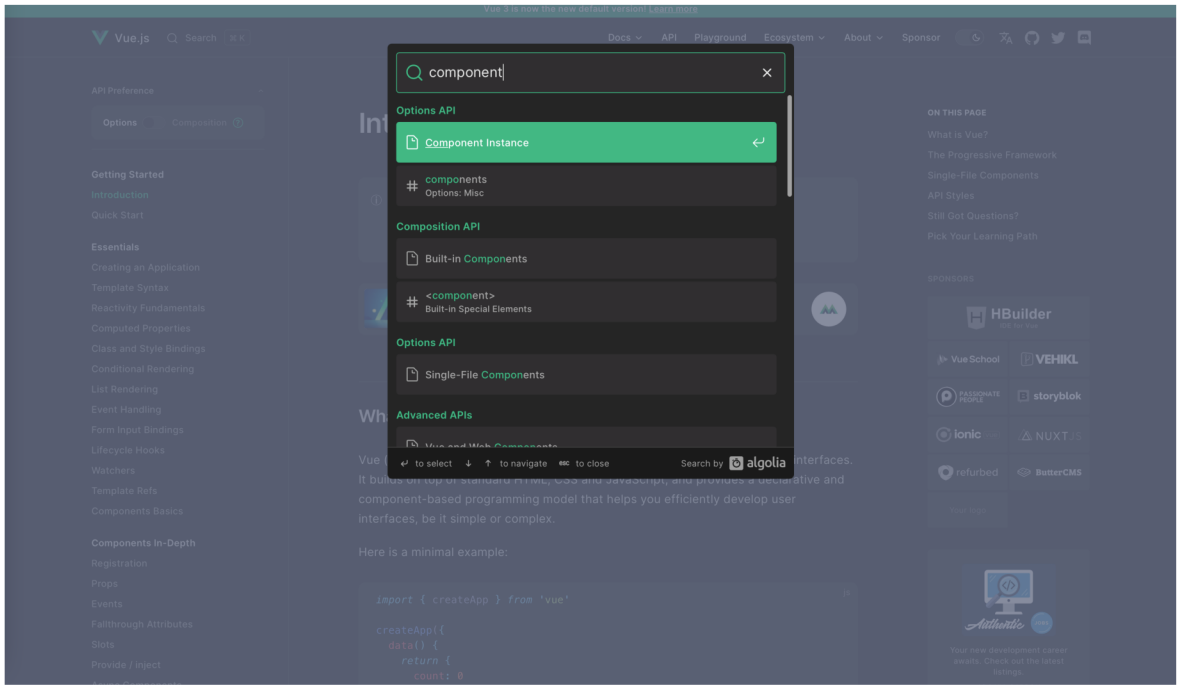
Příloha N - Statistiky subredditu r/angular2 [65]



Příloha O - Vyhledávání Component na angular.io



Příloha P - Vyhledávání Component na reactjs.org



Příloha Q - Vyhledávání Component na v3.vuejs.org