Porovnání JS frameworks podle specifik projektu

Jakub Petřík

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub Petřík**
Osobní číslo:	**A19088**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Porovnání JS frameworks podle specifik projektu**
Téma práce anglicky:	**Comparison of JS Frameworks According to Project Specifics**

Zásady pro vypracování

1. Nastudujte problematiku javascriptových frameworků.
2. Navrhněte a vyberte vhodné faktory pro volbu frameworku v kontextu typu aplikace a specifik projektu.
3. Vytvořte vhodné ukázkové aplikace dle vaší volby.
4. Vytvořené aplikace vhodně otestujte a porovnejte.
5. Vhodným způsobem reprezentujte výsledky práce.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. React. 07. září 2017. Scotts Valley, California, US: Createspace Independent Publishing Platform, 2017. ISBN 1976210232.
2. Vue: Step-By-Step Guide to Mastering Vue.Js from Beginner to Advanced. 07. září 2017. Scotts Valley, California, US: Createspace Independent Publishing Platform, 2017. ISBN 1976214386.
3. FREEMAN, Adam. Pro Angular 9: build powerful and dynamic web apps. Fourth edition. [New York, NY]: Apress, [2020]. ISBN 1484259971.
4. JavaScript Frameworks for Modern Web Development. 01. listopad 2019. New York City: APress, 2019. ISBN 1484249941.
5. Academind [online]. Grünwald: Maximilian Schwarzmüller, 2020 [cit. 2021-12-01]. Dostupné z: https://academind.com/tutorials/angular-vs-react-vs-vue-my-thoughts
6. Codeinwp [online]. Romania: Shaumik Daityari, 2021 [cit. 2021-12-01]. Dostupné z: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

Vedoucí bakalářské práce: **Ing. Petr Žáček, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 22. 5. 2022

Jakub Petřík v. r.
podpis studenta

ABSTRAKT

Bakalářská práce je zaměřena na porovnání JS frameworků dle zvolených faktorů v souvislosti se specifiky projektů. V rámci bakalářské práce bylo nejprve nutné představit čtenáři základní přehled o typech aplikací, popis JavaScriptu, TypeScriptu a jednotlivých frameworků. Dále pro vhodné porovnání a otestování je součástí práce navržena stejná multiplatformní aplikace, zaměřena na konkrétní typ projektu a vyvíjena 3 různými technologiemi. Aplikace je vhodně otestována a výsledky jsou vhodně odprezentovány.

Klíčová slova: JavaScript, Framework, multiplatformní aplikace, React, Angular, Vue

ABSTRACT

The bachelor thesis is focused on the comparison of JS frameworks according to the chosen factors in relation to the specifics of the projects. Within the scope of the bachelor thesis it was first necessary to present the reader with a basic overview of application types, a description of JavaScript, TypeScript and individual frameworks. Furthermore, for a suitable comparison and testing, the thesis includes the same proposed multi-platform application, focused on a specific project type and developed with 3 different technologies. The application is suitably tested and the results are suitably presented.

Keywords: JavaScript, Framework, multiplatform apps, React, Angular, Vue

Chtěl bych poděkovat vedoucímu mé bakalářské práce, panu Ing. Petru Žáčkovi, Ph.D. za ochotu, přínosné rady a pomoc při zpracování této práce.

OBSAH

ÚVOD

Svět technologií webových aplikací se v poslední době vyvíjí dost rychle a aplikace před 10 lety vypadají dneska hodně zastarale, pokud se nepředělávaly do moderního stylu. Nejenomže aplikace dneska vypadají lépe a moderněji, ale dostalo se nám i hodně možností jakým způsobem je vyvíjet. Vedle klasického HTML, CSS, JS se vyvinuly JS frameworky, které nabízí ulehčený způsob vývoje. Mezi nejznámější se řadí v dnešní době trojice React, Angular a Vue. Přínosem práce je seznámení čtenáře s jednotlivými technologiemi a následné pochopení, jak fungují, to vede k lepší orientaci mezi frameworky a podle výhod či nevýhod, ale hlavně faktorů si tak čtenář dovede určit, co je pro něho nejlepší.

Tato bakalářská práce se zabývá porovnáváním JS frameworků a následné vybrání toho, který je pro daný typ projektu ideální. Volba vhodného frameworku se určuje podle navrhnutých kritérií a faktorů, které mají nějaký dopad na samotnou aplikaci. Jako prezentace jsou zde vytvořené typy aplikací a na každou z nich je použita jiná technologie (JS framework). Tyto aplikace jsou následně řádně otestovány a porovnány takovým způsobem, aby člověk získal přehled o tom, jaký framework vybrat a proč.

Teoretická část zahrnuje vývoj multiplatformních aplikací, jejich rozdělení na nativní, webové a hybridní, dále se zabývá popsáním JavaScriptu a TypeScriptu, které jsou důležitou součástí frameworků, konkrétně jak se od sebe liší a taky je zde uvedeno, k čemu slouží API a DOM. Následuje nejdůležitější kapitola a sice JS frameworky, kde se nejdříve popisuje, co znamenají, na jakém principu fungují a jakým způsobem ulehčují vývoj. Potom se popisují konkrétní frameworky, jak vznikly, hlavní funkce a způsob fungování, nechybí zde Ionic framework, který je součástí vývoje aplikace. Poslední částí v teorii je porovnání JS frameworků, které vychází právě ze zmíněného popisu technologií, konkrétně se jedná o rozdíl v architektuře a syntaxi kódu, ale hlavní faktory jsou stanoveny až v praktické části.

V praktické části se důkladně popisuje API, která je použita na vykreslování obsahu do aplikace, její možnosti a jakým způsobem jsou data zobrazována. Další kapitola se věnuje popsání celé aplikace, jako první obsahuje přehled o tom, jak vypadá uživatelské prostředí a co se vlastně na každé stránce nachází. Nechybí však zde i podrobný postup vývoje aplikace, a hlavně programy a nástroje, bez kterých by se vývoj neobešel. Samotné testování je popsáno na třech typech aplikace a sice webová, Android jako nativní a PWA a jsou zde popsány problémy, se kterými se na dané platformě aplikace setkala. Poslední kapitola se

věnuje hlavnímu porovnávání JS frameworků podle předem stanovených faktorů a kritérií, jako příklad se popisoval rozdíl v rychlosti nebo princip samotného renderování a další.

I. TEORETICKÁ ČÁST

1 VÝVOJ APLIKACÍ

Aplikace se v dnešní době rozdělují na nativní a webové nebo je možnost vytvoření takzvané hybridní aplikace, která kombinuje tyto dva typy dohromady. Při vývoji požadujeme, aby aplikace byly multiplatformní, to znamená, aby aplikace byla podporována na platformách, jako je Android a iOS.

Poslední dobou se vývoj a design jednotlivých aplikací sjednocuje podle základních prvků dané platformy, aby aplikace nabízela stejný uživatelský zážitek a to, na co jsou uživatelé zvyklí. Tomu dopomáhá UI/UX design systém, podle kterého se aplikace navrhuje.

Vývoj aplikací je klíčový, protože to ovlivňuje většinu odvětví, aniž bychom si to uvědomovali a navíc čas, který trávíme ať už na mobilních telefonech nebo počítačích se zvyšuje. Díky tomu můžou vznikat tak skvělé způsoby, jak v dnešní době fungovat a nabízet tak jednoduché řešení každodenních problémů. Aplikace změnily způsob fungování hodně lidí, například takové Dáme jídlo nabízí hromadu stravovacích služeb, které Vám dovezou až ke dveřím, zatímco populární aplikace na poslech hudby Spotify nabízí obrovskou knihovnu všech skladeb, a to i offline.

1.1 Nativní aplikace

Nativní způsob vývoje aplikací je nejvíce používaný, protože aplikace je přímo vyvíjena na konkrétní typ platformy, a to vede k určitým výhodám/nevýhodám oproti ostatním způsobům. Nativní aplikace tak umožňuje mít větší výkon, to znamená, že zaručuje rychlou odezvu i při složitějších operacích. Další výhodou je přímý přístup k hardwaru zařízení, tím pádem je možné naprogramovat jakoukoliv komponentu a funkcionalitu aplikace, která by ostatními způsoby nešla, kvůli čekání na zpřístupnění určité API. Samozřejmostí je fungování aplikace v offline režimu, ale to už dneska nabízejí i PWA. Výhodou nebo nevýhodou může být distribuce aplikací pouze ze specifického obchodu dané platformy. Výhodou tohoto způsobu stažení je větší důraz na bezpečí a soukromí, kdy jsou aplikace kontrolovány předtím, než vstoupí do obchodu. Pokud však uživatel preferuje stažení aplikace přímo z webu, tak tohle bude brát jako nevýhodu. Nativní aplikace musí být vyvíjeny pro každou platformu zvlášť a tím pádem je vývoj náročnější, a hlavně práce se tím prodlouží, protože se musí programovat ve více programovacích jazycích, konkrétně u Androidu se jedná o Javu a Kotlin, u iOS byl využíván jazyk objektového C, ale dnes se píšou aplikace ve Swiftu. Dnešní moderní nástroje naštěstí dokážou tuto nevýhodu řešit tím,

že převedou napsaný kód jedním programovacím jazykem do nativního programovacího jazyku pro oba operační systémy. Mezi tyto nástroje patří React Native, Xamarin, NativeScript.

1.2 Webové aplikace

V poslední době vznikla možnost vytvářet aplikace přímo na webu, tento způsob nebylo možné ještě pár let dozadu realizovat, protože vývoj mobilních aplikací prostřednictvím JavaScriptu bylo pouze jenom experimentování. Kód webových aplikací je tvořen úplně stejným způsobem jako normální web, tedy HTML, CSS, JS a popřípadě JS frameworky. Díky tomu se jedná asi o nejjednodušší způsob, jak vytvořit aplikaci, a to nejen z pohledu vývoje, ale i nákladů. Protože se jedná o webovou aplikaci tak výkon je potom závislý na prohlížeči, na kterém se daná aplikace spustí a některé prohlížeče lépe spolupracují s JavaScriptem a některé zase hůře. Výhodou webových technologií však je, že aplikace je automaticky multiplatformní, tedy stačí vytvořit pouze jednu webovou aplikaci a tu je možno spustit na všech platformách, právě pomocí prohlížeče na daném zařízení. Webové aplikace můžou mít několik podob, jak má aplikace vypadat, jako první se nabízí řešení SPA (Single-page application), to znamená, že celý obsah stránky je tvořen pouze na jednu jedinou stránku a naviguje se pouze v ní. Nebo je možnost vytvoření takzvané PWA aplikace (Progressive web apps), která nabízí funkce jako aplikace nativní, tedy větší přístupnost k hardwaru zařízení. Jak již bylo zmíněno v kapitole 1.1 o nativních aplikacích, tak webovou aplikaci není nutné stahovat, a to může ulehčit vnitřnímu uložišti zařízení, protože se nenachází v žádném obchodu s aplikacemi. Tento způsob vývoje se nejčastěji využívá na malé a nenáročné aplikace, které nepotřebují pracovat s velkým počtem dat a mít přístup ke všem hardwarovým komponentám zařízení.

1.2.1 SPA

Je to vlastně jednostránková aplikace, jejíž kontent se nachází pouze na jedné hlavní stránce, tento způsob umožňuje skvělou orientaci uživatele nad obsahem dané stránky. Data jsou ukládána na straně klienta lokálně při načtení, tedy dochází k dynamickému načítání, tím dokáže být reakce okamžitá. Vývoj takových aplikací zprostředkovávají JavaScriptové frameworky a knihovny, jako například React, Angular, Vue.js a další. Samozřejmostí je i funkčnost bez připojení k internetu.

1.2.2 PWA

Jedná se o webovou aplikaci nebo stránku, která vypadá a funguje úplně stejně jako nativní mobilní aplikace. Toto řešení dává do budoucna hodně veliký smysl, a tak většina velkých firem přechází na PWA, protože s tím odpadávají nepříjemnosti, kdy se nebude muset aplikace aktualizovat z obchodu jako je Google Play nebo App Store, uživatelé totiž budou vždycky používat aktuální verzi dané aplikace.

1.3 Hybridní aplikace

Hybridní aplikace se dostávají více k popularitě, protože kombinují nativní a webové způsoby vývoje. K vývoji je zapotřebí pouze znát HTML, CSS a JS, podobně jako u webových aplikací, tím odpadá i znalost tvůrců aplikací, jelikož se nemusí učit nativní jazyk každého systému. Je zde možnost vyvíjet pomocí frameworků jako je Ionic, React Native atd., které vývoj značně urychlí, konkrétně Ionic framework má už v sobě definované komponenty, které se dají použít do každé aplikace. Největším problémem vývoje hybridních aplikací je fakt, že nemá přímý přístup k hardwaru, tím pádem, pokud neexistuje plugin, který by takové spojení umožnil, tak není možné, aby daná funkce v aplikaci fungovala. Ke každé hardwarové novince zařízení se musí tak čekat, než dojde k podpoře ze strany tvůrce frameworku. Na rozdíl od webových aplikací jsou hybridní výkonnější a je možnost je distribuovat pomocí obchodů. [1]

2 JAVASCRIPT

JavaScript se řadí mezi skriptovací jazyky, to znamená, že můžeme dávat jednotlivé instrukce dalším softwarovým programům, jako je webový prohlížeč nebo server. Jeho použitelnost se však nevztahuje pouze na web, ale umožňuje nám programovat i aplikace, které nejsou součástí webového prohlížeče. JavaScript tedy pokrývá jak front-endovou část, tak i back-end, tyto dvě části se dají ještě rozdělit na client-side, co vidí koncový uživatel nebo zákazník a server-side neboli to, co je pouze na straně serveru a uživatel k tomu nemá přístup. S příchodem nového JavaScriptového enginu V8 se vytvořilo back-endové multiplatformní prostředí Node.js, které funguje na principu spouštění JavaScriptového kódu právě mimo prohlížeč a pomocí něj píšeme celou programovou logiku na straně serveru.

Pokud nechceme, aby webová stránka, případně aplikace nebyla pouze statická, tedy nešlo s ní více manipulovat a nezobrazovala pouze statické informace, tedy kde nám nestačí webové technologie HTML, CSS tak potřebujeme použít právě zmíněný JavaScript, který zaručí složité funkce. Těmito funkcemi se myslí například dynamické načítání obsahu, změny týkající se informací, pokud chceme nějakou interakci s daným obsahem stránky nebo aplikace, ale taky i ověřování správných údajů zadaných ve formuláři nebo přihlášení [2].

Tohle všechno funguje tak, že v HTML se napíšou specifické tagy, nejčastěji tlačítko, které potřebuje interakci, aby se po kliknutí něco stalo

```
1  <button id="btn">Click</button>
```

Obrázek 2.1 HTML tag tlačítka

Tlačítko se za pomocí DOMu v JavaScriptu vezme

```
1  const btn = document.getElementById("btn");
```

Obrázek 2.2 Převzaté tlačítko z HTML do JS

Potom se na něho přidá event listener, který umožňuje zadat jakým způsobem se bude s daným objektem interagovat – za pomocí „click“ a jaká funkce bude použita k vykonání činnosti

```
1  btn.addEventListener("click", click);
```

Obrázek 2.3 Event listener na tlačítku

Jako poslední stačí naprogramovat logiku funkce, která bude aplikována na tlačítko a provede se

```
1  function click() {
2    console.log("Btn clicked...");
3  };
4
```

Obrázek 2.4 Funkce tlačítka

JavaScript patří do skupiny interpretovaných jazyků, to znamená, že kód se čte z vrchu dolů a výsledek kódu je okamžitě vracen, tedy kompilace se provádí za běhu. Ve skutečnosti používá just-in-time kompilování, což je metoda, při které je JavaScriptový kód kompilován do rychlejšího binárního formátu, aby mohl být spuštěn co nejrychleji [3].

2.1 Rozhraní API

Ještě větší využitelnost JavaScriptu spočívá v propojení s API, poskytující funkce nebo informace navíc, které by byly jinak velmi náročné naprogramovat. API je rozhraní pro programování aplikací, můžeme tak dostat do prohlížeče nebo aplikace data jednodušším způsobem.

Obecně se rozhraní API rozděluje na:

- **Rozhraní API prohlížeče** – z názvu je jasné, že jsou to data a funkce týkající se pouze prohlížeče, mezi takové patří zmíněná DOM API, Geolocation API, Canvas a WebGl a nechybí taky API pro zvuk a video pro interakci s multimédii
- **Rozhraní API třetích stran** – tohle jsou naopak rozhraní, která nejsou součástí přímo prohlížeče, ale musí se externě přenést ze zdrojů třetích stran. Například umožňuje vyjmout poslední příspěvky z Twitteru a zobrazit je na vlastní stránce.

Ve stručnosti se API dá představit jako databáze dat na straně klienta, která jsou pomocí JavaScriptu na určitou stránku nebo aplikaci zobrazovány [3].

2.2 DOM

Jak již bylo zmíněno v kapitole 2, tak JavaScript používá k přístupu k objektům metody DOM. DOM jako takový není programovací jazyk, ale bez něj by neměl JavaScript vůbec žádné ponětí o webových stránkách a nešlo by ho tak použít na tento způsob vývoje. Je to vlastně struktura dokumentu, nejčastěji HTML nebo XML představující webovou stránku v paměti, pro lepší orientaci a přístup. Lze tak upravovat jakýkoliv element v této struktuře, a to včetně stylů CSS. Objektový model dokumentu je nezávislý, a tak se dá použít i u jiných programovacích jazyků, když bude potřeba přístup k webovým položkám [4].

```
···<!DOCTYPE html> == $0
  <html lang="en">
  ▼<head>
     <meta charset="UTF-8">
     <meta http-equiv="X-UA-Compatible" content="IE=edge">
     <meta name="viewport" content="width=device-width, initial-scale=1.0">
     <title>Document</title>
   </head>
  ▼<body>
     <button id="btn">Click</button>
     <script src="main.js"></script>
     <!-- Code injected by live-server -->
   ▶<script type="text/javascript">…</script>
   </body>
  </html>
```

Obrázek 2.5 HTML Dokument

3 TYPESCRIPT

Už z názvu vyplývá, že se jedná o typovací jazyk, založený na JavaScriptu, který přináší možnost psát objektově orientovaný kód. Objektově orientovaný kód neboli OOP umožňuje psaní programové logiky ve třídách, a tak zlepšit čitelnost celého kódu. TypeScript byl naprogramován společností Microsoft a ta původně chtěla pouze přenést objektově orientované programy na web.

JavaScript nese s sebou řadu nevýhod, jednou z nich je nezjistitelnost chyby v kódu ještě před tím, než se program spustí, a tak si musíme až v prohlížeči, konkrétně v konzoli najít co je přesně špatně. Na takové chyby TypeScript upozorní už během psaní kódu. Syntaxe kódu je stejná a tím pádem nepovažuje kód JavaScriptu jako chybu, tedy přesunout JavaScriptový kód do TypeScriptu je možné a kód poběží beze změny díky zachování stejného fungování spuštěného programu, ale bude vyhazovat chyby už v editoru.

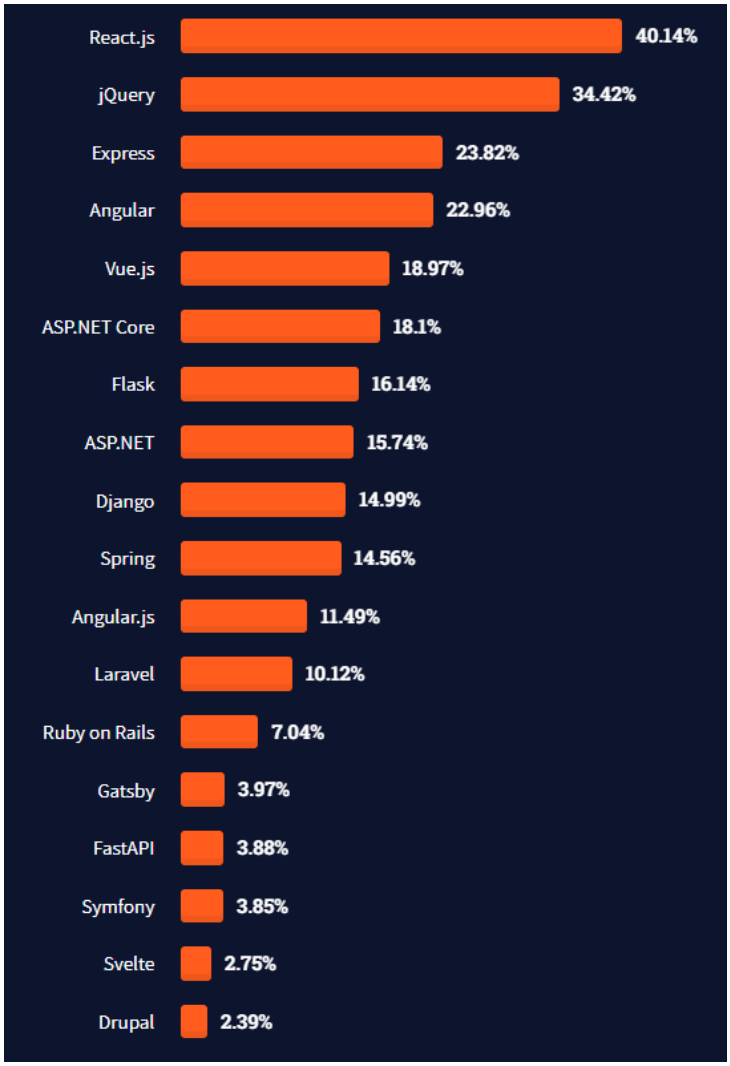
TypeScript přináší do JavaScriptu možnost přidávat jednotlivé typy proměnných a tím natvrdo určit o jaký typ se jedná. Špatné použití těchto proměnných vyústí potom v chyby. Nutno podotknout, že samotný prohlížeč nedokáže TypeScript spustit, tím pádem je zapotřebí kompilátoru, který daný kód přeloží do klasického JavaScriptu bez typových informací. To znamená, že během kompilace se mohou objevit chybové hlášky, ale nebude to mít žádný vliv na funkčnost spuštěného programu [5].

Obrázek 3.1 Ukázka kódu JS (vlevo) a TS (vpravo)

4 JAVASCRIPTOVÉ FRAMEWORKY

Hlavním tématem dnešní doby, jak vyvíjet multiplatformní aplikace jsou JS frameworky, které poskytují programátorovi jednodušší způsob vývoje. JS frameworky jsou tvořeny kolekcí JavaScriptových knihoven, které už někdo naprogramoval a díky tomu nabízí pestrou škálu užitečných funkcí, nachystaných pro programátory. Důležité je si uvědomit, že k vývoji aplikací vůbec nemusíme používat frameworky a celé to naprogramovat v klasickém JavaScriptu, ale takhle budeme muset při každém vývoji aplikace tvořit funkce od začátku, a to je dost neefektivní způsob. Proto tohle do značné míry řeší JS frameworky, ve kterých se programová logika funkce může sdílet a stavět tak na již existující sadě funkcí [6].

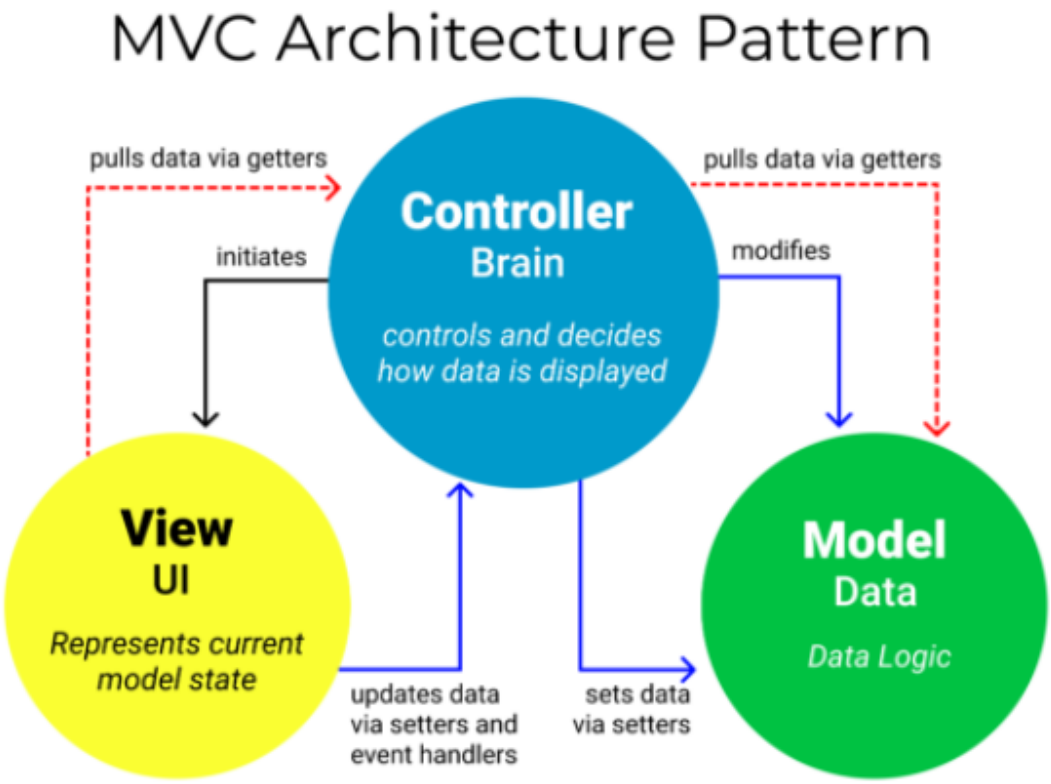
Výběr Reactu, Angularu a Vue jako JS frameworky pro tuto práci je z důvodu jejich rozšířenosti mezi programátory. Důležitým faktem, minimálně u prvních dvou je to, že za těmito frameworky stojí jedny z největších světoznámých společností a tím pádem je tady záruka relevance a aktualizací do budoucna, případně přizpůsobení se na dané požadavky programátorů a hlavně trhu. Dalším důvodem byla jedinečná kombinace těchto frameworků s Ionic frameworkem, který je použitý z důvodu lepší integrace a přizpůsobení se komponent na danou platformu. Níže se nachází graf 18-ti nejpoužívanějšími webovými frameworky za rok 2021, údaje pochází ze stránky Stack Overflow a na tento průzkum odpovědělo přes 67 000 lidí.



Obrázek 4.1 Graf nejpoužívanějších frameworků [7]

4.1 MVC

JS frameworky fungují na principu návrhového vzoru Model-View-Controller (MVC), který se běžně používá pro vývoj uživatelských rozhraní. Toto rozhraní se potom rozděluje do 3 vzájemně propojených prvků.



Obrázek 4.2 MVC architektura [8]

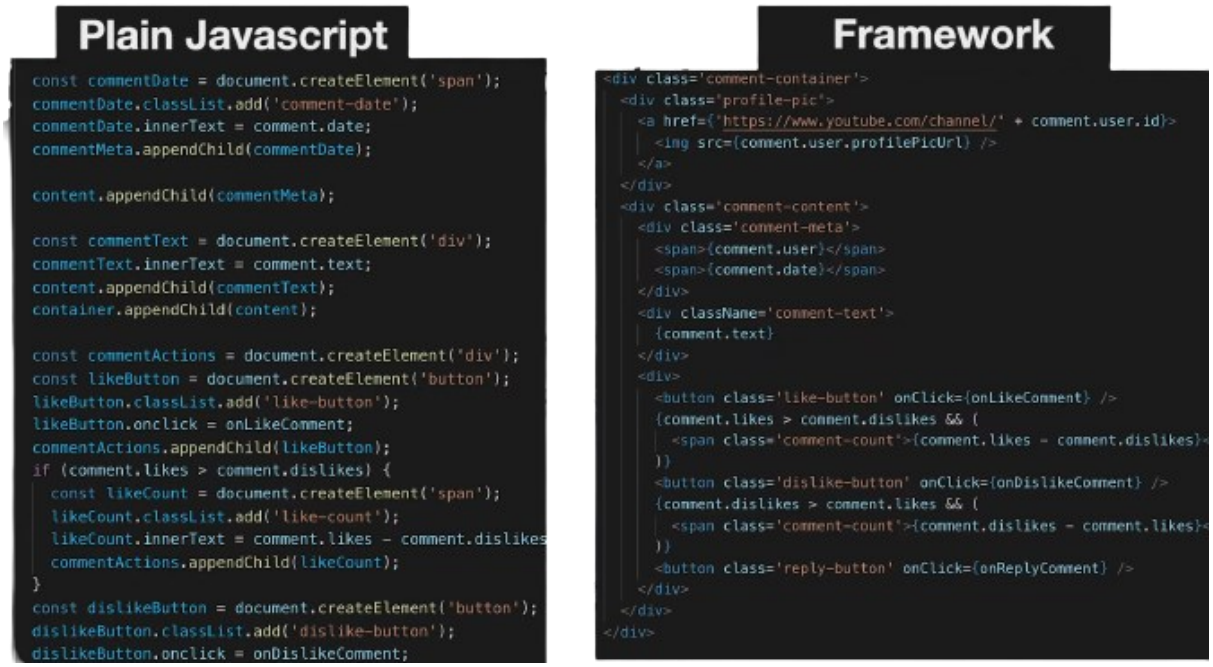
Model je důležitým prvkem pro webové komponenty, protože zajišťuje zpracování dat, ať už se jedná o data z databáze, API nebo JSON objektu [8].

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  render() {
    return (
      <div>
        <h1>Hello, world:</h1>
        <h2>It is {this.state.date.toLoca
      </div>
    );
  }
}
```

Obrázek 4.3 Model v Reactu [9]

View reprezentuje kód, který bude nějaký obsah vykreslovat, nejčastěji právě zmíněná data renderovat neboli zobrazovat na cílovou aplikaci nebo web. Ještě, než vznikly frameworky, museli vývojáři načítat složitější dynamická data pouze za pomocí JavaScriptu, u kterého to vypadalo velmi složitě a nepřehledně, zatímco u JS frameworků, konkrétně v této ukázce se jedná o React, který přináší HTML do JS kódu, a tak připomíná klasický HTML dokument [9].



Obrázek 4.4 Porovnání JS oproti Reactu [9]

Controller je interpreter, jehož úkolem je přijímat vstupy, které se následně upravují a poskytují upravená data zpátky uživateli. Má na starosti rozhodování o tom, jakým způsobem se budou data zobrazovat na modelu nebo view [8].

Obrázek 4.5 Event Handling – Controller [9]

4.2 Rozdíl mezi frameworkem a knihovnou

Nejčastějším příkladem diskuse bývá rozdíl mezi frameworkem a knihovnou. Plnohodnotný JS framework jako je například Angular obsahuje sadu nástrojů, které jsou využitelné pro budování stavby komplexní webové stránky nebo aplikace, zatímco JS knihovna je kolekce předem napsaných útržků kódu poskytující knihovnu funkcí dle potřeby [6].

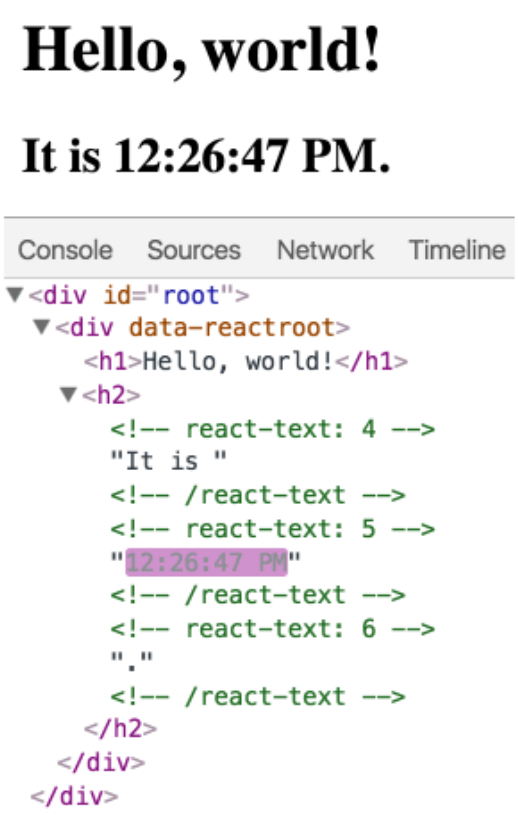
4.3 Router

Každá aplikace má spoustu komponent a každá z nich obsahuje svoje view. Je potřeba routeru, který když uživatel vyvolá nějakou akci, třeba zmáčkne tlačítko, které někam naviguje, tak, aby se požadavek splnil. V klasickém fungování navigace se musí HTML a JS dotázat serveru, aby se mohlo přejít na danou stránku. Odlišností routeru je to, že dokáže pracovat pouze na straně klienta, a tak ulehčit serveru.

Oproti klasickému linku (<a>) se vyplatí router používat k navigování mezi stránkami, protože klasický link má tu nevýhodu, že se stránka musí pokaždé opakovaně načíst, jinými slovy ztratí při navigování svůj aktuální stav, zatímco routing dokáže uchovávat svůj stav a tím nepotřebuje načítat stránku.

4.4 React

React vzniknul Facebookem (dneska pod názvem Meta) v roce 2013 jako reakce na nedostatek tehdejších technologií, protože neexistovaly rozšířené možnosti, jak vyvíjet uživatelské rozhraní aplikace. Tato JavaScriptová knihovna je založena na principu skládání jednotlivých dílčích komponent do sebe, a tak vytvářet strukturu celé aplikace. Komponenty potom můžou přijímat vstupní data a mít svůj vlastní stav, který když se změní, tak se proběhlé změny vykreslí. Soubory, které obsahují programovou logiku a zároveň vrací upravenou komponentu se píšou v JSX nebo TSX, ale pokud si chceme například pouze definovat pole prvků, které budeme dále používat, tak stačí použít klasický JS nebo TS, podle toho, s jakým jazykem zrovna pracujeme. React si dokáže udržovat aktuální stav DOMu a porovnat změny s novým stavem, který byl změněn, to znamená, že dokáže upravit pouze prvky, které se stavem změnily nebo přidaly a zbytek komponenty nechá být v původním stavu, tím se nemusí aktualizovat celý DOM a provedení změn je tak rychlejší [10], [11].



Obrázek 4.6 Měnící se jeden prvek v DOMu [12]

4.4.1 JSX

Jedná se o kombinaci psaní HTML a JS kódu, abychom dostali přehlednější šablonu s využitím JavaScriptu naplno. Zkratka vznikla spojením JavaScriptu a XML a znamená to umožnění psát HTML tagy přímo do JavaScriptového kódu. React vás samozřejmě nenutí používat JSX, ale kvůli lepší přehlednosti kódu se ho vyplatí používat, aby kód nevypadal složitěji a na první pohled člověk nevěděl o co vlastně v programu jde.

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

root.render(<HelloMessage name="Taylor" />);
```

Obrázek 4.7 Kód s použitím JSX [11]

```
class HelloMessage extends React.Component {
  render() {
    return React.createElement(
      "div",
      null,
      "Hello ",
      this.props.name
    );
  }
}

root.render(React.createElement(HelloMessage, { name: "Taylor" }));
```

Obrázek 4.8 Kód bez použití JSX [11]

Důležitou poznámkou je, že kvůli tomu, že JSX má blíže k JavaScriptu tak React DOM používá camelCase vlastnosti místo klasických HTML atributů, příkladem může být použití className pro určení třídy prvku, na rozdíl class.

Nevýhodou je, že tento kód neumí prohlížeč přečíst, proto se musí zavést kompilátor JavaScriptu zvaný Babel, který převede daný kód na jednoduchý kód JS [13].

4.4.2 Komponenty

Jedním z hlavních stavebních kamenů Reactu jsou komponenty, tedy nezávislé, znovupoužitelné React elementy se svojí funkcionalitou, které se dají použít kdekoliv v projektu a skládají tak komplexní aplikaci. Tyto komponenty pak obsahují svoje vlastnosti (props), které se dají představit jako parametry funkce, stejně jako v libovolném programovacím jazyku, ale na rozdíl od nich pracují se vstupními daty a vrací prvek React. Ačkoliv je React dost flexibilní, má svoje striktní pravidla, kdy se tyto „props“ nemůžou upravovat ve vlastní komponentě a zapisují se do React elementů, stejně jako atributy. Komponenty se dají napsat pomocí funkce nebo pomocí třídy ES6.

Dlouho se používaly třídy, protože nabízely funkcionality jako použití stavu nebo životního cyklu, které ve funkcích použít nešly. Díky představení dalších funkcích JavaScriptu zvaných React Hooks tato nevýhoda u funkcí odpadá a tím i zjednodušuje problémy s logikou psaní ve třídě, například potřeba psaní this [14].

4.4.3 Hooks

React Hooks přišly do Reactu od verze 16.8 a řeší problematiku spojenou s opakovaným použitím chováním té samé komponenty, to znamená že pomocí Hooks je možné vyjmout stavovou logiku z komponenty a tu potom následně nezávisle testovat nebo použít v dalších komponentách, aniž by se měnila její struktura. Dřív se totiž musela pomocí návrhových vzorů měnit struktura celé komponenty, a to nebylo ideální.

Mimo to tak Hooks usnadňují práci s rozdělováním komponent na menší funkce, podle toho, jak spolu souvisejí a nevynucují tak rozdělování na základě metod životního cyklu.

React Hooks jsou vlastně přídavné funkce Reactu, které nabízí stav a životní cyklus ve funkcích. Konkrétně životní cyklus použitý ve třídách býval rozprostřen do několika metod, a to na přehlednosti zrovna moc nepomáhalo [15].

4.4.3.1 Effect Hook

React Hooks nabízí použití useEffect, který v sobě obsahuje tři momenty životního cyklu komponenty, mezi které patří zavedení do DOMu (componentDidMount), přerenderování (componentDidUpdate) a případně odebrání z DOMu (componentWillUnmount). UseEffect v sobě kombinuje tyto metody a sjednocuje je do jednoho rozhraní API.

Obrázek 4.9 Rozdíl mezi životním cyklem třídy (vlevo) a React Hooks (vpravo)

Ve zkratce se obsah useEffect volá při každém načtení dané komponenty, ale je možné taky specifikovat po jaké změně se useEffect zavolá znovu, například při změně nějakého stavu [16].

4.4.3.2 State Hook

Další funkce pro práci se stavy se nazývá useState, kterou využíváme pouze, když chceme změnit počáteční stav nějakého objektu, pole, čísla nebo řetězce. Výhodou stavu oproti normální proměnné je ten, že se stav zachovává i po skončení funkce. UseState tedy vrací dvě hodnoty, aktuální stav, ve kterém se nachází a funkce, pomocí které se dá ten stav změnit [17].

4.5 Angular

Angular framework je open-source, který je vyvíjen společností Google, využívá se na velké komplexní projekty, na které už nevystačí pouze HTML, CSS a JS. Stejně jako React se v Angularu sestavují komponenty, které jsou následně používány v aplikaci nebo webu. Je postavěn na jazyce TypeScript, hlavním principem bude tedy objektově orientované programování a poskytuje sbírku integrovaných knihoven, aniž bychom museli nějaké navíc instalovat. Pokud bychom však chtěli upřednostnit JS, můžeme použít předchůdce Angularu a to AngularJS, který už z názvu napovídá, že se jedná o JavaScriptový framework. Přičemž v dnešní době je více oblíbený a používaný novější Angular s TypeScriptem, protože se dost podobá zmíněnému konkurentovi Reactu a hlavně nabízí vyšší výkon a rychlost. Důležité je zmínit, že tyto dvě „verze“ Angularu nejsou mezi sebou kompatibilní, protože využívají jiné principy a jazyk [18], [19].

4.5.1 Komponenty

Komponenty v Angularu se skládají z HTML šablony, která říká, co se má na danou stránku nebo aplikaci zobrazit, potom TypeScript třídu, ve které se píše celá programovací logika dané komponenty a důležitou součástí je CSS selektor, který definuje, jakým způsobem se má komponenta použít v šabloně, v neposlední řadě nesmí chybět soubor pro CSS styly.

```
@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
```

Obrázek 4.10 Angular TS soubor komponenty

Každá komponenta obsahuje HTML šablonu, která může být definována v odděleném souboru nebo se dá napsat přímo do programové logiky, tohle se ale většinou používá na malé komponenty s menším obsahem tagů.

Angular taky rozšiřuje možnosti, jak vkládat dynamické hodnoty z komponenty do jazyka HTML, kde roli hraje aktuální stav, který když se změní, tak se automaticky aktualizuje vykreslený DOM.

Pro interakci mezi rodičovskou komponentou a podřízenou komponentou se používají takzvané „Input dekorátory“, které slouží jako vstupní hodnoty, jejichž úkolem je sdílet data mezi komponentami [18], [20].

4.5.2 Lifecycle Hooks

I Angular poskytuje programátorovi možnost používat Hooks metody pro potřebu upravit nějakou hodnotu, pokud došlo k její inicializaci nebo změně či odstranění. Umožňuje na tyto změny událostí reagovat a ve vhodnou chvíli s nimi manipulovat. Hooks se dají představit jako sekvence metod volající se v určitém pořadí, které reagují na určité změny týkající se stavu nebo hodnot.

Každé rozhraní definující jako metodu pro Hooks obsahuje předponu ng, příkladem může být rozhraní OnInit, které má Hooks metodu s názvem ngOnInit(). Tato metoda se volá krátce po tom, co byly zkontrolovány vstupní hodnoty dané komponenty nebo directives.

Angular samozřejmě obsahuje více takových metod, které slouží pro jiné účely, další z nich může být ngOnChanges() metoda, která se volá před zmíněným ngOnInit() a reaguje na

nastavení nebo resetování vstupních vlastností (Input dekorátory), které jsou vázané na poskytnutá data [21].

4.5.3 Directives

Directives jsou vlastně upravené HTML atributy, které rozšiřují chování prvků. Nejčastěji se používají k jejich modifikaci, tedy změně stylu nebo chování elementu celkově. Může se tak za pomocí použití NgClass na prvek měnit třída podle určité proměnné typu boolean, která vrací true nebo false. Tyto directives lze využít i přímo v metodě, kde se nabízí více možností, jak s výslednou hodnotou manipulovat pomocí programové logiky.

Důležité je také použití NgModelu, který se stará o zobrazování vlastností dat a jejich aktualizace, když uživatel provede nějaké změny.

Mnohem standardnější je využití NgIf, která funguje úplně stejně jako normální podmínka, NgFor, která prochází celý list [22].

4.5.4 Dependency Injection

Architektura celého Angularu funguje na principu komponent a na využití návrhového vzoru Dependency Injection, který se stará o konkrétní část aplikace, konkrétně služby, které jsou poskytnuty třídě, aby mohla plnit své funkce. Tyto služby jsou totiž mimo komponentu, tedy potřebuje se k nim dostat z externích zdrojů a jsou předávány pomocí konstruktoru třídy [23].

4.6 Vue

Dalším JavaScriptovým frameworkem je Vue.js, který byl napsán tehdejším čínským programátorem v Google Evana You v roce 2013. Google však nejevil o tento framework zájem, a tak se Evan rozhodl nechat pozice a naplno se věnovat této technologii. On jako samotný považuje Vue jako takový „nástupce“ Angularu, protože se snažil vzít jeho části a vytvořit něco ještě víc odlehčeného, a hlavně něco, co usnadňuje práci.

Vue se řadí mezi progresivní frameworky, tedy ty, které se do budoucna budou přizpůsobovat změnám ve webových technologiích, a hlavně potřebám programátorů, proto nabízí velkou škálu funkcí, jakým způsobem dnes tvořit web nebo aplikaci.

Stejně jako u Angularu, tak Vue rozšiřuje HTML syntaxi, aby nám umožňoval zpracovat výstup na základě JavaScriptového stavu, ve kterém se právě nachází. Dokáže pomocí

reaktivnosti sledovat aktuální stav a když nastane změna, tak Vue efektivně aktualizuje DOM [24], [25].

4.6.1 Komponenty

Vue komponenta je založena na Single-File, to znamená, že celá komponenta je psaná pouze do jednoho souboru, podobnému HTML s koncovkou vue. V tomto souboru tedy nalezneme jak celou strukturu šablony HTML a CSS, tak i samostatnou programovou logiku naprogramovanou v JavaScriptu, případně TypeScriptu. Tyto bloky obsahují svoji značku, kam se zapisují, kvůli přehlednosti.

Obrázek 4.11 Uvnitř Vue komponenty

Tento způsob byl použit hlavně kvůli lidem, kteří s webovým vývojem teprve začínají, aby syntaxe byla co nejpodobnější obyčejnému vývoji HTML, CSS, JS.

Komponenty podobně jako u Reactu využívají „props“ pro předávání dat mezi nimi.

Vue ale nabízí možnost vytvářet tyto komponenty dvěma různými styly API. Prvním stylem je Options API, který definuje komponentu pomocí objektů, jako jsou data a metody, případně metoda životního cyklu (Lifecycle Hooks). Tento způsob připomíná objektově orientované programování ve třídách.

```
data() {
  return {
    count: 0
  }
},
```

Obrázek 4.12 Options API [24]

Dalším stylem je Composition API, který importuje potřebné funkce do komponenty, které se potom využívají pro definování stavu a použití Hooks. Pro tuto možnost je potřeba znát principy reaktivnosti, ale jakmile je znáte, tak nabízí větší flexibilitu a schopnost opakovaně využít tuto logiku [25], [26].

```
import { ref, onMounted } from 'vue'

// reactive state
const count = ref(0)
```

Obrázek 4.13 Composition API [24]

4.6.2 Reaktivita

Jednou z důležitých funkcí Vue, ale i ostatních frameworků je reaktivita, tento systém umožňuje řešit problémy se změnou dynamických dat jednoduchým způsobem, který by v obyčejném JavaScriptu vyžadoval složitější operace.

```
let A0 = 1
let A1 = 2
let A2 = A0 + A1

console.log(A2) // 3

A0 = 2
console.log(A2) // Still 3
```

Obrázek 4.14 Nelze změnit výsledek po přiřazení jiné hodnoty [27]

Podstatnou roli hraje stav komponenty, který když se změní tak se ta změna projeví a aktualizuje případné zobrazení [27].

4.6.3 Lifecycle Hooks

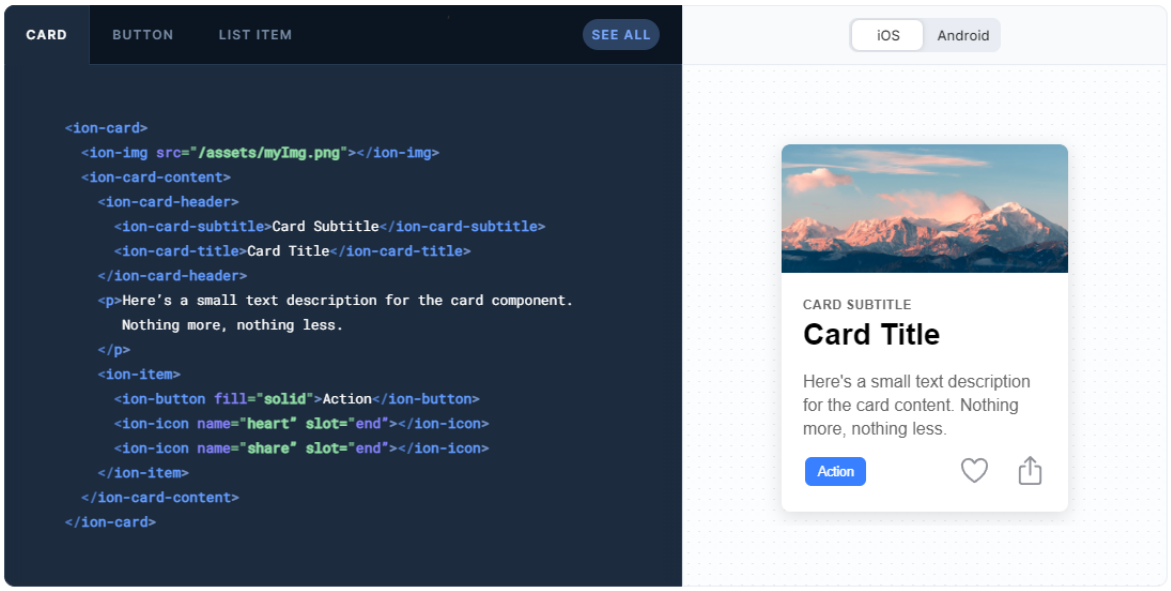
Při tvorbě komponenty musí její obsah projít několika kroky, než dojde k jejímu vykreslení na web nebo aplikaci. Hooks specifikují programátorovi metody, které se při této cestě mohou spouštět, aby zajistily potřebnou funkci zdrojového kódu. Pro použití například nejčastější metody, která se spustí po úspěšném načtení komponenty je potřeba pouze napsat mounted() do script bloku [28].

4.7 Ionic framework

Ionic je open-source sada nástrojů pro vývoj softwaru neboli SDK napsaný v JavaScriptu, jeho předností jsou hybridní mobilní aplikace. Vyvinula jej společnost Drifty Co. v roce 2013 a v té době byl postaven na platformě AngularJS a Apache Cordova. Ionic je framework, který se vyplatí používat hlavně pro mobilní aplikace, ale dá se pomocí něj sestavit i webová aplikace, její vývoj bude ovšem složitější kvůli responzivitě ionic komponent. Snaží se cílit na to, aby vývoj mobilních aplikací byl jednodušší tím, že programátor napíše pouze jeden základ kódu a ten se pak přizpůsobí dané platformě (Android, iOS, Windows). Tento způsob usnadňuje tvoření multiplatformních aplikací. Ionic tedy nabízí komponenty, které potřebují programovou logiku na to, aby fungovaly podle našich představ a k tomu lze využít právě JS frameworky, případně je možné použít pouze obyčejný JavaScript [29], [30].

4.7.1 Komponenty

Všechny Ionic komponenty jsou postaveny na základních webových technologiích, jako je HTML, CSS a JavaScript, které využívají moderní Web API a Shadow DOM. Jednotlivé komponenty jsou navrženy tak, aby vypadaly stejně jako nativní komponenty a snaží se dodržet i designové prvky, animace a gesta typické pro danou platformu, nechybí však možnost upravit si komponentu podle svých představ.



Obrázek 4.15 Příklad Ionic komponenty [31]

Aby se daly považovat za plnohodnotné nativní komponenty, tak jim nechybí nativní přístup k funkcím nebo vlastnostem daného zařízení, tedy mají přístup k fotoaparátu, GPS nebo Bluetooth a dalším funkcím s pomocí nativních pluginů, které dodávají Capacitor nebo starší Cordova [32].

4.7.1.1 Shadow DOM

Jak již bylo zmíněno výše, tak komponenty jsou tvořeny základními webovými technologiemi, které využívají DOM k přístupu k jednotlivým objektům. Právě Ionic používá Shadow DOM k tvoření vlastních komponent. Shadow DOM tedy umožňuje připojit „skrytý DOM“ do klasického DOM elementu, tím získáme řadu dalších elementů v jedné komponentě. Příkladem může být obyčejný přehrávač videa, který má v HTML značku <video>, která v sobě obsahuje prvky jako spustit video nebo dát video na celou obrazovku atd… Výhodou je, že Shadow DOM je úplně oddělený, tím pádem obecné modifikace stylů nebo JavaScriptu bez použití určité funkce k přístupu do Shadow DOM nebudou mít na danou strukturu vliv, ale hlavně aby se zachovala čistota a přehlednost kódu a nepletly se vzájemné části [33].

4.7.2 Capacitor a Cordova

Capacitor představuje multiplatformní rozhraní, které má za úkol přenést webovou aplikaci do nativní i s přístupem k hardwarovým. Vytváří tak pojem webové nativní aplikace, to

znamená webová aplikace, která se chová stejně jako nativní, a to tím způsobem, že Capacitor poskytuje sadu rozhraní API, aby aplikace podléhala webovým standardům, ale zároveň šla nativně spustit na dané platformě. Jednotlivé nativní funkce se zpřístupňují přes Plugin API pro programovací jazyky spojené s konkrétním systémem, iOS využívá Swift, Adroid zase Javu a klasický web používá JavaScript.

Jako alternativu lze využít předchůdce Capacitoru Apache Cordova, který v podstatě nabízí stejné funkce pro spuštění webové aplikace na nativní platformě, ale s tím rozdílem, že se nespustí jako progresivní webová aplikace. Výhodou Cordovy je ale to, že byl vyvinut už v roce 2009 a tak obsahuje velikou škálu rozhraní, a tak přístup téměř ke všemu. Capacitor na rozdíl od Cordovy byl vydán v roce 2018 a poskytuje tak mnohem modernější rozhraní API, které ale nemusí obsahovat přístupy ke všem nativním prvkům [34], [35].

5 TEORETICKÉ POROVNÁNÍ JS FRAMEWORKŮ

5.1 Architektura

Prvním porovnáním je z hlediska architektury, jakou architekturu využívá daný framework a jak to ovlivňuje jejich fungování kódu.

Začneme Reactem, který nespadá přímo do MVC architektury, ale to neznamená, že by ji nešlo použít, jen k tomu nebyl stvořen. React používá plnou sílu programovacího jazyka, který potom vykresluje správné zobrazení, tedy využívá pouze složku View z celého MVC. Psaním kódu v Reactu popisujeme, co se bude vykreslovat ve výsledné aplikaci nebo webu.

Předchůdce AngularJS se stavěl na architektuře MVC, ale kvůli nevýhodě, která neumožňovala aktualizaci modelu, když se provedla změna ve view se s nástupem Angularu předělala na architekturu MVVM (Model View View-Model). Controller je tedy nahrazen View-Modelem, který je ve skutečnosti funkce JavaScriptu, která má na starosti podobně jako controller starat se o udržování vztahu mezi view a modelem s tím rozdílem, že pokud změníme něco ve view složce, proběhne úprava i v modelu a pokud změníme něco v modelu, zobrazí se to ve view. Této metodě se ještě říká 2-way binding. Angular se do teď dá považovat i jako MVC architektura, kvůli oddělení všech těchto částí od sebe [36].

Vue kvůli tomu, že z velké části vychází z Angularu, tak využívá MVVM architekturu, který odděluje uživatelské rozhraní od logiky aplikace. Model-View je tu zase jako zprostředkovatel mezi controllerem a view [37].

5.2 Syntaxe kódu

V této sekci si řekneme o rozdílech ve psaní zdrojového kódu v jednotlivých JS frameworcích, jak se zapisuje logika.

U Reactu, jak již bylo zmíněno v kapitole 4.4, používáme takzvaný JSX způsob, který nám umožňuje psát normální HTML tagy do JavaScriptového kódu, aniž bychom ztratili jeho funkčnost. To nám dává možnost lepší manipulace s tím, co se bude zobrazovat. Pokud chceme psát JavaScriptový kód je zapotřebí ho obalit do složených závorek {}, což je speciální syntaxe JSX pro vyhodnocení výrazu při kompilaci. Takovým výrazem může být proměnná, funkce, objekt nebo jakákoliv úprava kódu [38].

```
{item.backdrop_path ? (
  <IonImg
    src={"http://image.tmdb.org/t/p/original/" + item.backdrop_path}
  />
) : null}
<div className="ion-margin-bottom">
  {item.genres?.map((item: any) => {
    return <IonChip key={item.id}>{item.name}</IonChip>;
  })}
</div>
```

Obrázek 5.1 React syntaxe

Angular používá šablonu na psaní toho co se má zobrazit, tedy klasické HTML a oproti Reactu rozšiřuje tyto elementy o JavaScriptovou logiku. Této logice se říkají directives, které se zapisují jako atributy a umožňují tak například procházet položky v poli, pro vypsání vlastností těchto položek potom slouží dvojité složené závorky {{}}. Dá se taky pomocí hranatých závorek [] nastavovat hodnoty těchto vlastností, cílem je vlastnost DOM, které se přiřadí daná hodnota, například zdroj obrázku [39].

```
<ion-img *ngIf="backGroundImage" [src]="backGroundImage"> </ion-img>
<div class="ion-margin-bottom">
  <ion-chip *ngFor="let item of genres">{{item.name}}</ion-chip>
</div>
```

Obrázek 5.2 Angular syntaxe

Vue taky používá šablonu a vlastně umožňuje to samé jako Angular s tím rozdílem, že má jinou syntaxi atributů (directives), například v-if pro podmínku. Na nastavování dynamických hodnot potom používá v-bind, které se může zkrátit na pouhou dvojtečku (:id), která je považována jako validní znak pro atribut. Pokud je tato hodnota null nebo není definována tak se tento atribut z elementu odstraní [40].

```
<ion-img v-if="movie.backdrop_path" :src="imageUrl + movie.backdrop_path"></ion-img>
<div class="ion-margin-bottom">
  <ion-chip v-for="genre in movie.genres" :key="genre.id.value">{{
    genre.name
  }}</ion-chip>
</div>
```

Obrázek 5.3 Vue syntaxe

5.3 Vlastní stanovené faktory

Prvním zvoleným faktorem je rychlost, její porovnání se nachází v kapitole 9.1.1 a je to parametr, který je velmi důležitý při tvorbě každého softwaru. Uživatel totiž nebude rád, když načítání a celé fungování aplikace je pomalé a zasekané, problém nastává i v případě, kdy se aplikace načítá pár sekund, protože udržení pozornosti je klíčové a pokud uživatel musí čekat už pár desítek sekund, tak odchází z webu nebo aplikace. Kvůli tomuto důvodu se musí rychlost software neustále testovat, aby si udržel stálou návštěvnost a dobré hodnocení.

Následuje kapitola 9.1.2, kde se porovnává velikost projektu a tento faktor je zde hlavně kvůli lepšímu orientování se v celém projektu, a to platí i u přehledu o tom, jaký obsah se v daném souboru nebo složce nachází. Někomu může více vyhovovat komponenta rozdělená do několika souborů, kdy má oddělenou logiku od šablony a někomu zase vyhovuje všechno v jednom souboru.

Dalším faktorem je množství vloženého obsahu v kapitole 9.1.3, kde je potřeba porovnat jaká je velikost bez obsahu a pak s obsahem, tedy co bylo nutné doinstalovat navíc a velikost projektu se tak může celkem dost navýšit. Taky se zde získává přehled o tom, kolik je potřeba importovaných položek pro „jednoduchou“ komponentu.

Vytváření a renderování komponent s „props“ je faktor porovnávaný v kapitole 9.1.4 a řadí se mezi nejdůležitější faktory. Samotné vytváření komponent je v každém frameworku odlišné a je to základní stavební kámen při práci s nimi, proto je dobré vědět, jak náročné je takovou komponentu vytvořit. Předávání dat mezi komponentami jako „props“ funguje taky na jiném principu a každá technologie má svoje pravidla jakým způsobem to provádět. Součástí toho je renderování neboli vykreslování výsledného obsahu, které ovlivňuje

spoustu věcí, jako je dříve zmíněná rychlost načtení obsahu nebo při nějaké změně, za jak dlouho se určitá změna projeví a hlavně, aby se vše zobrazilo správně a funkční.

Posledním probíraným faktorem v kapitole 9.1.5 je rozdíl renderování na straně klienta a na straně serveru, které se v každém frameworku zpracovává jinak a na to, aby aplikace běžela na serveru je potřeba vždy jiného nástroje. Je tak možné zefektivnit načítání obsahu, tak aby uživatel něco viděl, než se načte zbytek, protože část kódu běží na serveru, který načte statický obsah hned.

II. PRAKTICKÁ ČÁST

6 THE MOVIE DATABASE API

Na aplikaci byla použita API ze stránky The Movie Database, která v sobě obsahuje velké množství filmů, seriálů a všech spojených informacích o nich, také obsahuje detaily o společnostech, které za daným snímkem stojí.

6.1 Seznam filmů

Každé zavolání požadavku na kolekci filmů jich může vrátit maximálně pouze 20 s tím, že u většiny se tento seznam neustále mění podle toho, k čemu je určený. Tento počet se dá změnit přechodem na jinou stránku, který obsahuje zase jiné filmy a na tomto příkladu jde vidět, že se dá vybírat až do 1000 stránek, tím pádem se dá dostat až k 20 000 filmům.


```
▼{page: 1,…}
   page: 1
  ▼results: [{genre_ids: [14, 28, 12], title: "Doctor Strange in the Multiverse of Madness",…}, {,…},…
    ▶0: {genre_ids: [14, 28, 12], title: "Doctor Strange in the Multiverse of Madness",…}
    ▶1: {,…}
    ▶2: {video: false, vote_average: 6.1,…}
    ▶3: {poster_path: "/7qop80YfuO0BwJa1uXk1DXUUEwv.jpg", video: false, vote_average: 7.8,…}
    ▶4: {video: false, vote_average: 7.2,…}
    ▶5: {id: 778106, original_language: "en", original_title: "Along for the Ride",…}
    ▶6: {poster_path: "/74xTEgt7R36Fpooo50r9T25onhq.jpg", video: false, vote_average: 7.8,…}
    ▶7: {original_language: "pl", original_title: "365 Days: This Day",…}
    ▶8: {,…}
    ▶9: {adult: false, backdrop_path: "/iQFcwSGbZXMkeyKrxbPnwnRo5fl.jpg", genre_ids: [28, 12, 878],…}
    ▶10: {adult: false, backdrop_path: "/192vHmAPbk5esL34XMKZ1YLGFjr.jpg", genre_ids: [28, 53, 80], ic
    ▶11: {adult: false, backdrop_path: "/4rON0QFWpHtZ9bEtfdE68TFbfGr.jpg", genre_ids: [53, 27],…}
    ▶12: {,…}
    ▶13: {,…}
    ▶14: {adult: false, backdrop_path: "/eQN31P4IEhyp6NkdccvppJnyuJ4.jpg", genre_ids: [28, 12, 14, 878
    ▶15: {adult: false, backdrop_path: "/AcStyrENNAzGB7hkCPe1YWgFgWM.jpg", genre_ids: [27], vote_count
    ▶16: {original_language: "en", id: 949581, poster_path: "/25MkSPKzlqbFwXhGU3KFm92WbL1.jpg", video:
    ▶17: {genre_ids: [35], title: "40 Years Young", original_language: "es", original_title: "Cuarento
    ▶18: {backdrop_path: "/figlwUsXXFehX3IebdjqNLV6vWk.jpg", genre_ids: [28, 53], original_language: "
    ▶19: {id: 454626, vote_average: 7.4,…}
   total_pages: 1000
   total_results: 20000
```


Obrázek 6.1 Výsledky trendy filmů

Existují však zde funkce, jak dostat mnohem více výsledků, největší je za použití discover, která vrátí sice taky 20 filmů, ale s tím, že počet stránek může dosáhnout přes 33 000.


```
▼{page: 209, results: [,…], total_pages: 33463, total_results: 669242}
   page: 209
  ▶results: [,…]
   total_pages: 33463
   total_results: 669242
```


Obrázek 6.2 Vrácení discover požadavku

6.2 Informace o filmu

Důležitým aspektem každého filmu jsou informace potřebné k jeho popsání, abychom věděli, co všechno obsahuje. Podle těchto dat se dá taky daný druh filmu vyfiltrovat a najít shodu.

Kromě základních informací jako jsou žánry, název, popis, datum vydání a hodnocení tak tato API poskytuje možnost zjistit, jestli je film pouze pro dospělé, filmové studio, které za snímkem stálo a jakých film dosáhl tržeb v kině nebo z prodeje Blu-Ray, případně obsahuje složku stav, ve kterém se daný film nachází, jestli už vyšel nebo ne.


```
▼{adult: false, backdrop_path: "/5P8SmMzSNYikXpxil6BYzJ16611.jpg",…}
  adult: false
  backdrop_path: "/5P8SmMzSNYikXpxil6BYzJ16611.jpg"
 ▶belongs_to_collection: {id: 948485, name: "The Batman Collection", poster_path: "/tuzKA9K5Ae9IzaA0
  budget: 185000000
 ▶genres: [{id: 80, name: "Crime"}, {id: 9648, name: "Mystery"}, {id: 53, name: "Thriller"}]
  homepage: "https://www.thebatman.com"
  id: 414906
  imdb_id: "tt1877830"
  original_language: "en"
  original_title: "The Batman"
  overview: "In his second year of fighting crime, Batman uncovers corruption in Gotham City that cor
  popularity: 5709.032
  poster_path: "/74xTEgt7R36Fpooo50r9T25onhq.jpg"
 ▶production_companies: [{id: 101405, logo_path: null, name: "6th & Idaho", origin_country: "US"},…]
 ▶production_countries: [{iso_3166_1: "US", name: "United States of America"}]
  release_date: "2022-03-01"
  revenue: 764573021
  runtime: 176
 ▶spoken_languages: [{english_name: "English", iso_639_1: "en", name: "English"}]
  status: "Released"
  tagline: "Unmask the truth."
  title: "The Batman"
  video: false
  vote_average: 7.8
  vote_count: 4336
```


Obrázek 6.3 Data o filmu

6.3 Dodatečné informace

Bohužel v základních informacích chybí seznam herců, režisérů a ostatní co se na tvorbě filmu podíleli, a tak je potřeba dalšího požadavku API pro získání těchto dat.


```
▼{id: 453395, cast: [,…],…}
 ▶cast: [,…]
 ▶crew: [{adult: false, gender: 2, id: 531, known_for_department: "Sound", name: "Danny Elfman",…},…]
  id: 453395
```


Obrázek 6.4 Data o obsazení filmu

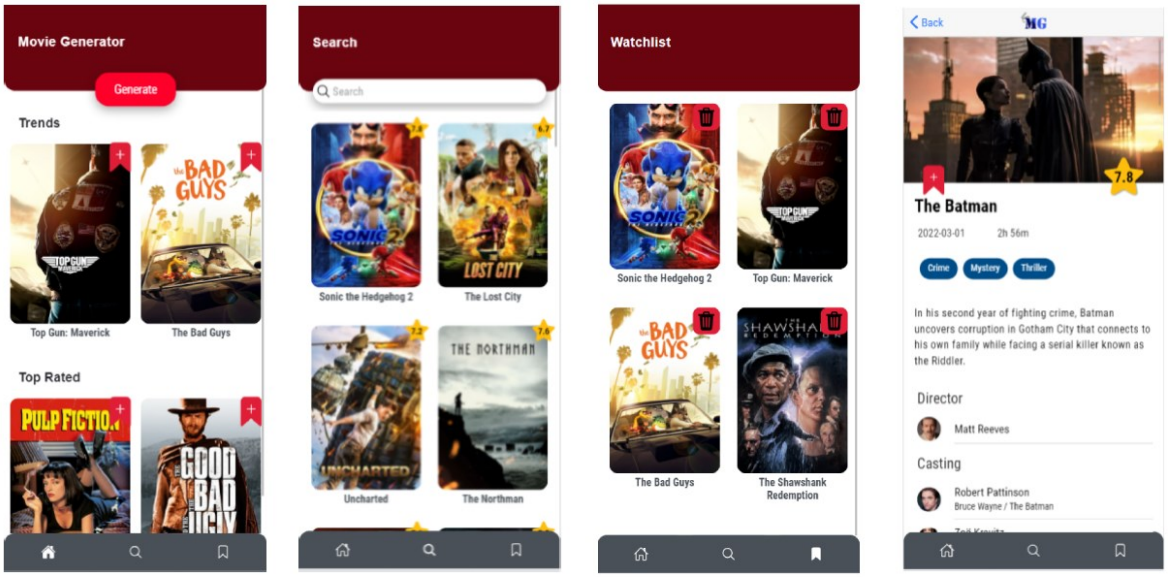
Tohle ale nepokryje vše, co je možné o filmu vědět, je zde také možnost přidat upoutávku na film, kterou je vhodné dat buď na stránku samotného filmu nebo na hlavní stránku jako navnadění na snímek. Další možností je přidání specifických klíčových slov, případně zobrazit seznam podobných filmů, které jsou stejného žánru nebo obsahují stejná klíčová slova. Nesmí však chybět uživatelské recenze, které jsou nedílnou součástí každého hodnocení filmu.

7 MOVIE GENERATOR APLIKACE

Zvolil jsem si aplikaci, jejíž primárním účelem je generování náhodných filmů. Tato aplikace nabízí uživateli přehled o filmech, které se na dané API nachází a v případě, pokud uživatele daný snímek zaujal tak si ho může uložit do watchlistu pro pozdější přehrání. K tomu dopomáhá rozšíření filmu o další informace, jako je jeho hodnocení, informace o hercích atd. Pro lepší přehlednost je zde také možnost vyhledávání konkrétního filmu, aby byly poskytnuty všechny nástroje vedoucí ke správnému výsledku. Důležitou součástí je tmavý režim, který je vytvořen na přizpůsobení se právě zvolenému režimu zařízení.

7.1 Uživatelské rozhraní aplikace

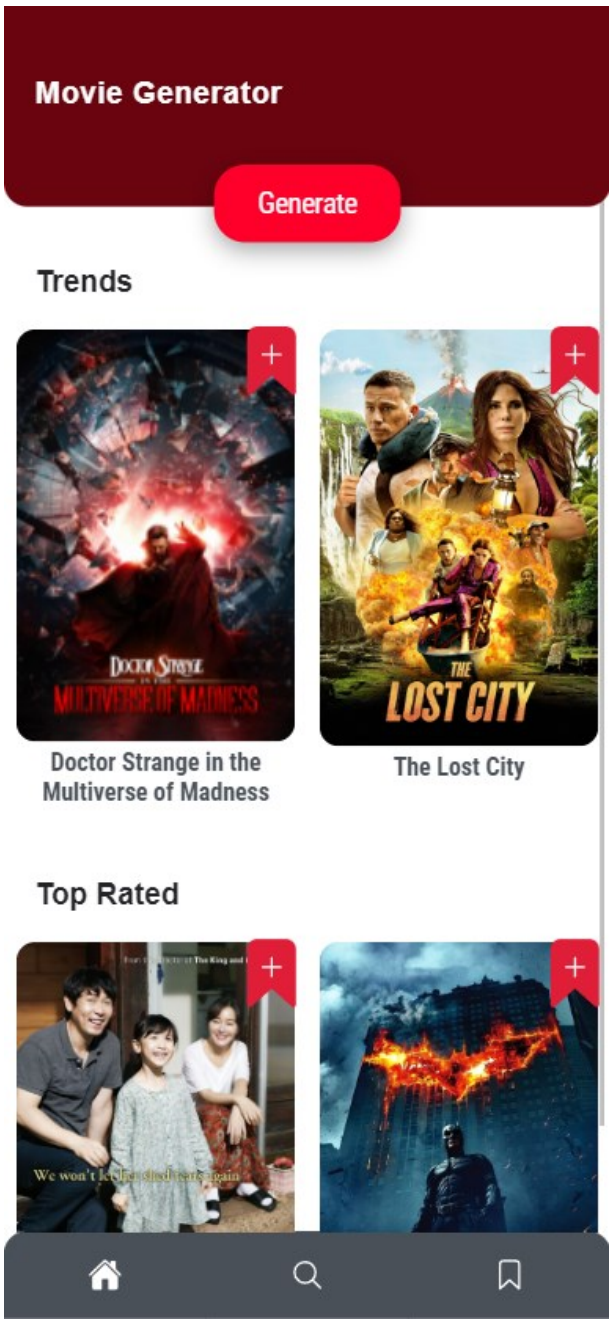
Aplikace je rozdělena do 4 stránek, z nichž hlavní stránky v navigaci obsahují stejný design horní části, kromě jednoho důležitého prvku a na všech stránkách jsou k dispozici navigační karty (tabs) ve spodní části. Hlavní část aplikace se mění v důsledku toho, pro jakou funkci daná stránka slouží.



Obrázek 7.1 Přehled uživatelského rozhraní aplikace

7.1.1 Titulní stránka

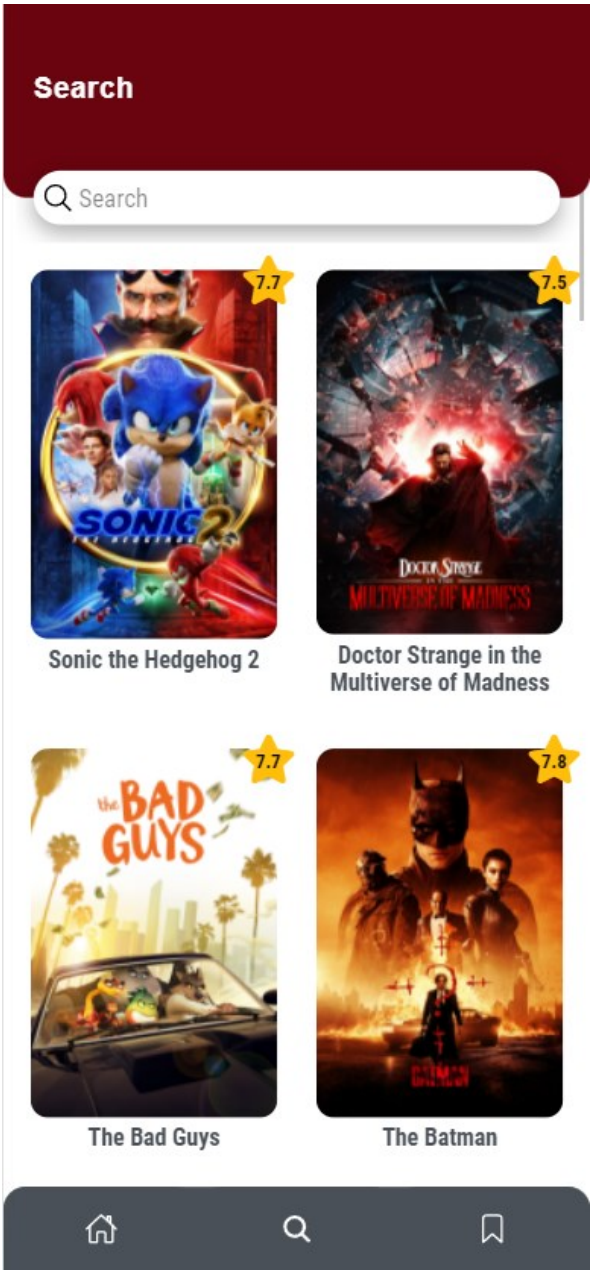
Na titulní stránce lze vidět hlavní tlačítko v hlavičce sloužící pro generování náhodných filmů. V hlavní části se nachází 2 slidery, které nabízí různé seznamy filmů s možností si ho přidat do watchlistu.



Obrázek 7.2 Titulní stránka aplikace

7.1.2 Vyhledávací stránka

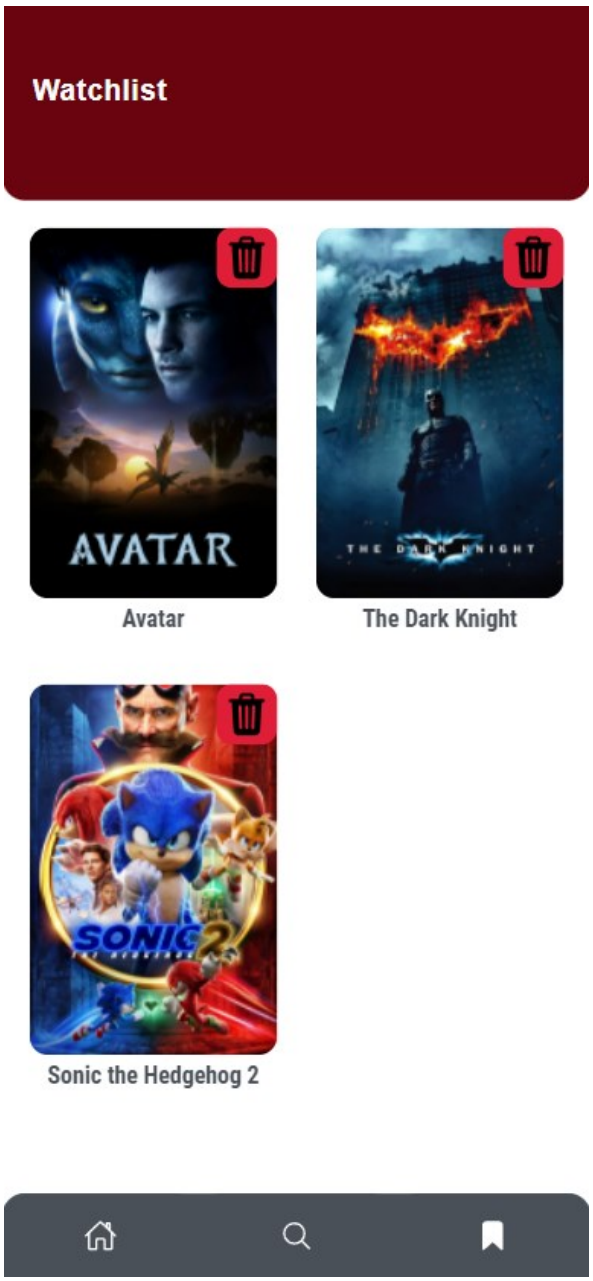
Tato stránka obsahuje v horní části vyhledávací pole pro zadání názvu filmu, který chce uživatel najít. Pod tímto prvkem se nachází nejdříve filmy, které jsou právě teď populární a po zadání nějakého názvu filmu do vyhledávače se tento seznam změní a seřadí podle nejlepšího možného výsledku. V tomto obsahu se dá taky posouvat dolů a nahoru pro další výsledky.



Obrázek 7.3 Vyhledávací stránka

7.1.3 Stránka sledovaných filmů

Na této stránce se nachází v hlavním obsahu seznam filmů, které byly přidány uživatelem do watchlistu, tyto položky je možné z tohoto seznamu sledovaných filmů odstranit, například pokud si je uživatel už přehrál nebo je tam už z nějakého důvodu nechce.



Obrázek 7.4 Watchlist stránka

7.1.4 Detailní stránka filmu

Detailní stránka o každém filmu obsahuje v horní části tlačítko s možností vrátit se na předchozí stránku a ve středu je logo aplikace, které odkazuje na hlavní stránku. Hned pod tím se nachází ikonický obrázek daného filmu, jeho součástí je přidání filmu do watchlistu a hodnocení na pravé straně, následuje název filmu a další informace spojené se snímkem od data vydání, žánrů, popisu až po samotného režiséra a celého hereckého obsazení.

Obrázek 7.5 Detailní stránka filmu

7.2 Postup vývoje aplikace

V této kapitole se čtenář dozví, jakým způsobem byla aplikace vyvinuta. Kroky níže mají za úkol podrobnější popsání postupu práce, konkrétně v této ukázce se jedná o kombinaci Ionic frameworku s Reactem.

7.2.1 Nástroje potřebné k vývoji

Nejdřív bylo zapotřebí si stáhnout potřebné programy k vývoji a nastavit je tak, aby vše fungovalo správně a bylo na čem stavět samostatnou aplikaci.

7.2.1.1 Node Package Manager (NPM)

Jako první je nutností si stáhnout Node Package Manager, který slouží jako správce balíčků pro JavaScript běžící na Node.js. Jelikož aplikace běží na JS frameworcích tak je potřeba těchto balíčků pro ulehčení práce s tvorbou celého projektu. Pomocí něho se dá doinstalovat potřebné balíčky do projektu nebo dokonce je možné tvořit vlastní balíčky a publikovat je veřejnosti. Pro instalaci npm je potřeba jít na stránky https://nodejs.org/en/ a stáhnout si nejlépe doporučenou verzi Node.js, úspěšnou instalaci je možné ověřit příkazem **npm -v** v příkazovém řádku na zjištění, jestli vše funguje.

7.2.1.2 Visual Studio Code

Aby bylo možné někde psát kód, tak potřebujeme nějaký textový editor. Programovat je možné samozřejmě v obyčejném poznámkovém bloku, ale je to neefektivní a přichází se o řadu vylepšení. Aplikace byla napsaná v editoru Visual Studio Code od Microsoftu, který nabízí lepší přehlednost kódu pomocí barev, automatického formátování kódu a také je možnost doinstalovat si potřebná rozšíření pro specifický programovací jazyk ve kterém se právě vyvíjí. Navíc umožňuje programátorovi si zapnout vestavěný terminál jako náhradu za příkazový řádek a mít tak vše na jednom místě.

7.2.1.3 Google Chrome

Pro spuštění aplikace nebo její testování je nutností mít nainstalovaný webový prohlížeč. Na tento vývoj byl použit Google Chrome, kde největší roli hrály nástroje pro vývojáře. Tato sekce nabízí všelijaký přehled o aplikaci, její kód, výkon, požadavky na síť, případně co uchovává za data. Nejvíc času bylo však stráveno v záložce konzole, která zobrazuje

vyskytlé chybové hlášky a taky hodně slouží jako takový debugger, tedy ladění celé aplikace do funkčního stavu.

7.2.1.4 Android studio

Důležitou součástí po tvorbě aplikace bylo testování, ke kterému se používalo Android studio, aby se dalo zjistit, jestli aplikace běží nativně na zařízeních s Androidem v pořádku. Můžete ho nainstalovat z přímo z oficiálních stránek https://developer.android.com/studio, ale mnohem složitější bylo jeho zprovoznění, musíte tak mít nainstalovanou Javu a potom si tyto dvě cesty nastavit v prostředí systému Windows, a to jak k Android studiu, tak k Javě.

7.2.1.5 Ionic CLI

Jako poslední důležitou součástí je možnost používat příkazy spojené s Ionic frameworkem. K tomu je zapotřebí instalace rozhraní příkazového řádku pro aplikace Ionic (Ionic CLI). Následující příkaz nainstaluje toto rozhraní globálně do počítače, a to umožní využívat ionic příkazy v jakémkoliv terminále.

```
npm install -g @ionic/cli
```

7.2.2 Vytvoření aplikace

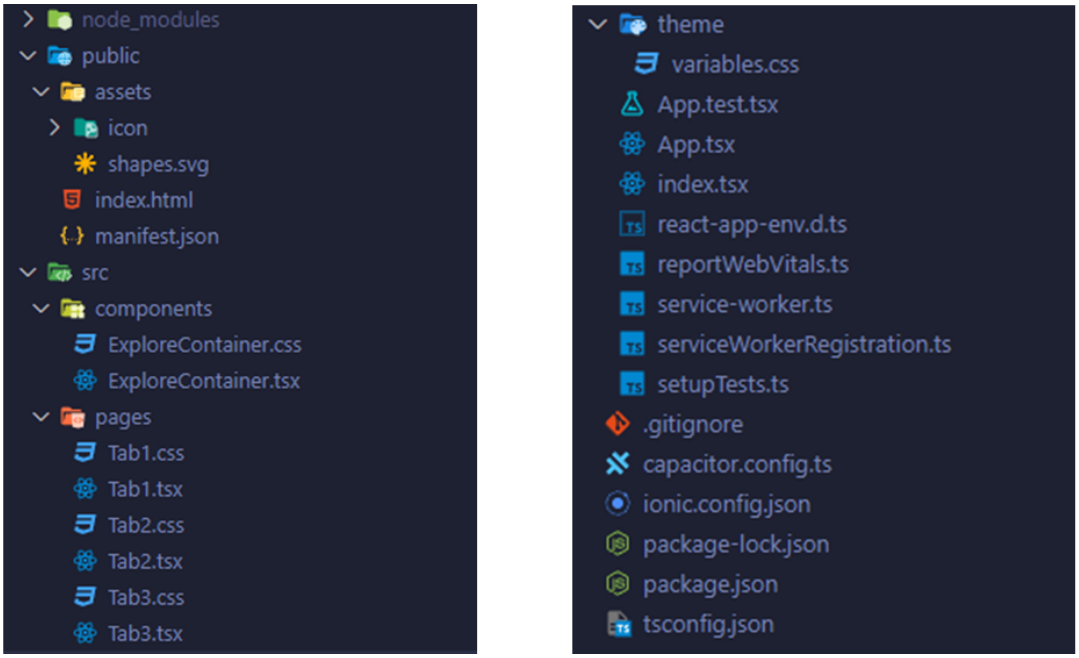
Ještě, než se bude aplikace vytvářet, tak se specifikuje cesta, kde se má složka s projektem vygenerovat a v této cestě se spustí následující příkaz pro vytvoření Ionic aplikace:

```
ionic start mg-app-react tabs –type=react –capacitor
```

V tomto příkaze se taky objevuje tabs, který předpřipraví šablonu se 3 navigačními záložky. Po chvilce strpení se vygeneruje projektová struktura, kterou si je potřeba otevřít ve Visual Studio Code.

7.2.2.1 Projektová struktura

Zde se nachází složka s projektem, ve kterém jsou potřebné soubory pro fungování aplikace.



Obrázek 7.6 Struktura projektu

7.2.2.2 Použití SASS

Ve výše uvedené struktuře lze vidět, že ve výchozím stavu se používá na styly CSS, ale pro lepší a přehlednější psaní stylů je potřeba změnit CSS na SASS. Tento krok je velice jednoduchý bude stačit nainstalovat node-sass balíček pomocí příkazu:

```
npm i node-sass
```

Potom je potřeba změnit všechny koncovky souboru z .css na .scss a upravit importy.

7.2.2.3 Sestavení route

Další postup je sestavení route tak, aby jednotlivé linky odkazovaly na správnou stránku. Všechny stránky jsem si sepsal do jednoho souboru .ts, ze kterého se potom dynamicky načítaly. Tento soubor obsahoval hlavně informace o stránkách, například jejich url cestu, komponentu, která se má zobrazit anebo ikonu na jednotlivé taby.

```
export const pages = [
  {
    path: "/home",
    icon: homeOutline,
    selectedIcon: home,
    component: Home,
    redirect: true,
  },
```

```
<IonTabs onIonTabsWillChange={(e) => setSelectedTab(e.detail.tab)}>
  <IonRouterOutlet>
    {pages.map((page, index) => {
      return (
        <Route
          path={page.path}
          component={page.component}
          key={index}
          exact
        />
      );
    })}
    <Route exact path="/">
      <Redirect to={pages.filter((page) => page.redirect)[0].path} />
    </Route>
  </IonRouterOutlet>
  <IonTabBar slot="bottom">
    {pages.map((page, index) => {
      if (page.icon) {
        const isSelected = selectedTab === `tab${index}`;
        return (
          <IonTabButton
            tab={`tab${index}`}
            href={page.path}
            key={index}
          >
            <IonIcon
              icon={isSelected ? page.selectedIcon : page.icon}
            />
          </IonTabButton>
        );
      }
    })}
  </IonTabBar>
</IonTabs>
```

Obrázek 7.7 Sestavení Route

7.2.2.4 API zpracování požadavků

Následovalo vytvoření samostatné složky services, uvnitř se nachází soubor ApiConnect.ts, který v sobě za pomocí JavaScriptové knihovny axios umožňuje provádět http požadavky přímo z Node.js. Pro použití je potřeba tuto knihovnu nainstalovat příkazem npm install axios a následně ji importovat. Potom už je možné vytáhnout data z URL API, pomocí jednotlivých funkcí.

```
import axios from "axios";
const apiUrl = "https://api.themoviedb.org/3";
const apiKey = process.env.REACT_APP_TMDB_API_KEY;

export const getTrendingList = () => {
  return axios
    .get(`${apiUrl}/trending/movie/day?api_key=${apiKey}&language=en-US`)
    .then((response) => {
      return response.data;
    });
};
```

Obrázek 7.8 Zpracování požadavků API

7.2.2.5 Implementace funkcí

Důležitou součástí aplikace jsou funkce a jejich implementace.

Generování náhodných filmů

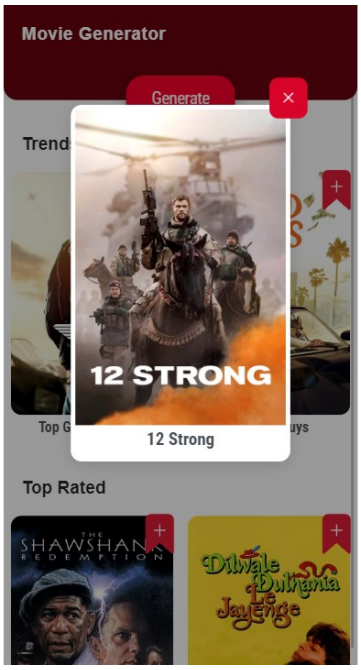
Jako první bylo potřeba zajistit fungování náhodného generování filmů, tato funkce vypočítává náhodnou stránku a film, které se používají jako argumenty pro získání dat z API. Tato funkce se následně volá na stisknutí tlačítka, které zobrazí modal okno s daným filmem.

```
const [modalMovie, setModalMovie] = useState<any>([]);
const [showModal, setShowModal] = useState(false);

const getRandomMovie = () => {
  const page = Math.floor(Math.random() * (500 - 1) + 1) + 1;
  const movie = Math.floor(Math.random() * 19);

  getDiscoverList(page).then((data) => {
    setModalMovie(data.results[movie]);
  });

  setShowModal(true);
};
```



Obrázek 7.9 Generování náhodných filmů

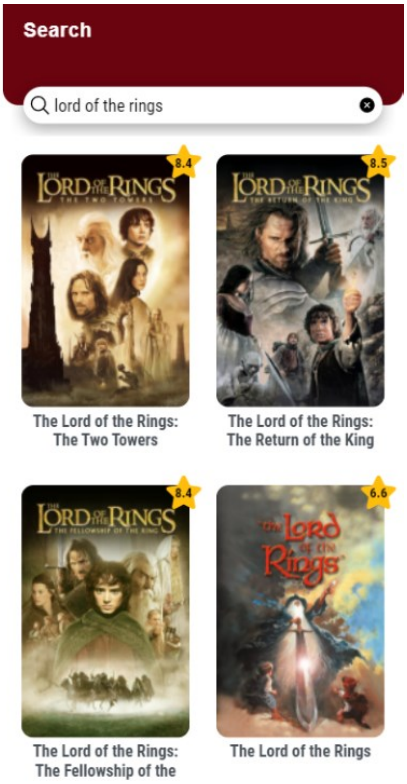
Vyhledávání

Samotné vyhledávání není tak složité, funguje vlastně na principu filtru, kdy uživatel napíše něco do vyhledávacího pole a obsah se na základě toho změní. V tomto případě je ale potřeba kontrolovat, jestli se tam nějaký text nachází, protože pokud ne tak se zobrazí seznam populárních filmů.

```
const filterData = () => {
  if (searchText.length !== 0) {
    loadSearchContainer();
  } else {
    getPopularMovies();
  }
};

const loadSearchContainer = () => {
  getSearchList(page, searchText).then((r) => {
    if (page > 1) {
      setItem([...item, ...r.results]);
      console.log("ahoj");
    } else {
      setItem([...r.results]);
    }
  });
};

const getPopularMovies = () => {
  getPopularList(page).then((r) => {
    if (page > 1) {
      setItem([...item, ...r.results]);
    } else {
      setItem([...r.results]);
    }
  });
};
```



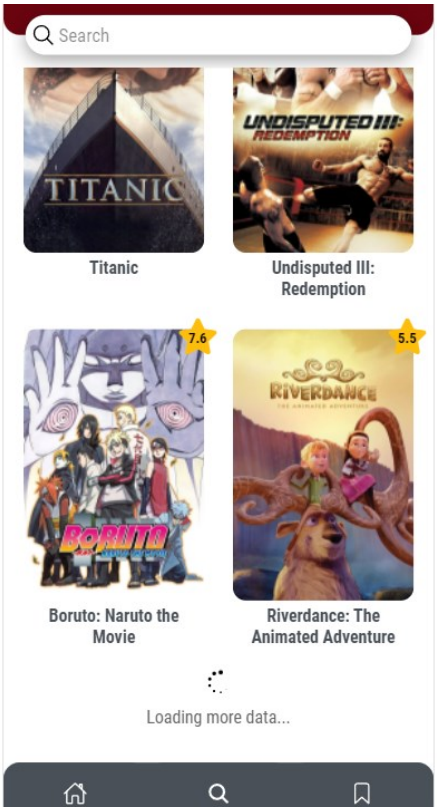
Obrázek 7.10 Vyhledávání

Infinite scroll

Tato funkce slouží k tomu, aby se daly najít všechny výsledky vyhledávání, tedy jakmile se dojde na konec seznamu (po 20 položce), tak se přičte stránka a načte se dalších 20 položek, dokud se nedojde na konec výsledků.

```
const searchNext = (event: any) => {
  setPage(page + 1);

  if (searchText.length !== 0) {
    loadSearchContainer();
  } else {
    getPopularMovies();
  }
  event.target.complete();
};
```



Obrázek 7.11 Infinite scroll

Přidání do local storage

Následuje funkce, která má za úkol získat všechna data o položce, kterou chce uživatel přidat, tato data se potom uloží do local storage a nesmí však chybět kontrola proti duplikaci stejných položek.

```
const getItems = (data: any) => {
  const items: any[] = [];
  if (JSON.parse(localStorage.getItem("items") || "[]") === null) {
    items.push(data);
    localStorage.setItem("items", JSON.stringify(items));
  } else {
    const localItems = JSON.parse(localStorage.getItem("items") || "[]");
    localItems.map((details: any) => {
      if (data.id !== details.id) {
        if (items[data.title] === undefined) {
          items[data.title] = data.title;
        }
        items.push(details);
      }
    });
    items.push(data);
    localStorage.setItem("items", JSON.stringify(items));
  }
};
```

Obrázek 7.12 Přidání do local storage

Odstranění z local storage

Tato funkce je hodně podobná té předchozí, akorát se musí zjišťovat kromě položky taky index, aby se dalo zjistit jaká položka se má z local storage odstranit. Pokud je seznam prázdný tak se celé local storage vyčistí.

```
const removeItem = (e: any, i: number) => {
  const items: any[] = [];
  var array = [...listLocalStorage];
  JSON.parse(localStorage.getItem("items") || "[]").map((data: any) => {
    if (data.id !== e.id) {
      items.push(data);
    } else {
      array.splice(i, 1);
      setListLocalStorage(array);
    }
  });
  localStorage.setItem("items", JSON.stringify(items));
  if (items.length === 0) {
    localStorage.clear();
  }
};
```

Obrázek 7.13 Odstranění z local storage

7.2.2.6 Nastavení PWA

Ještě, než se bude aplikace předělávat na PWA tak je nutné aplikaci nasadit na doménu a zpřístupnit hosting. K tomu byl použit firebase, který tyto možnosti umožňuje dost jednoduchým způsobem, stačí zbuildit aplikaci následujícím příkazem, aby obsahovala složku, která se bude zobrazovat obsah na webu.

ionic build –prod

Potom je potřeba si na https://firebase.google.com/ založit účet nebo použít účet od google a vytvořit nový projekt. Na konec se musí inicializovat aplikace na firebase pomocí příkazu firebase init a přiřadit daný projekt a příkazem firebase deploy nasadit.

7.2.2.7 Nastavení Android studia

Pro to, aby bylo možné aplikaci spustit nativně v Android studiu a moct tak testovat na emulátoru nebo reálném zařízení s Androidem, je potřeba nejprve doinstalovat capacitor pokud chybí a až potom se musí přidat do projektu android složka:

npm install @capacitor/core ionic capacitor add android

8 TESTOVÁNÍ APLIKACE

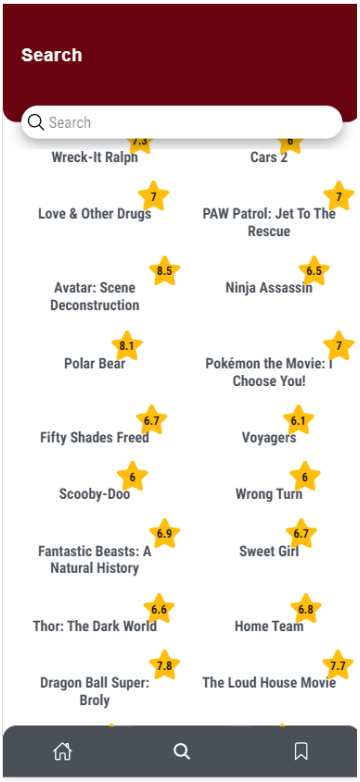
Tato kapitola se bude věnovat testování aplikace na různých zařízeních, či prostřednictvím různých programů.

8.1 Webová aplikace

Nejdřív kontrola probíhala na webu přes prohlížeč, kdy je aplikace hostována pomocí firebase. Odkaz na aplikaci: https://mg-app-fb0aa.web.app/

Na hlavní stránce byly testovány slidery a tlačítka, jestli fungují, jak mají a vypadá to, že všechno funguje v pořádku, akorát při rychlém skrolování slideru jde vidět, že se zasekne na nějakém filmu, ale to bude tím, že se nestačil načíst další seznam filmů.

Vyhledávací stránka trochu pomaleji načítá obsah, pokud se skroluje pomocí tzv. infinite loaderu až dolů, ale jinak vyhledávání je bleskově rychlé filmy se mění rychle.



Obrázek 8.1 Načítací obsah

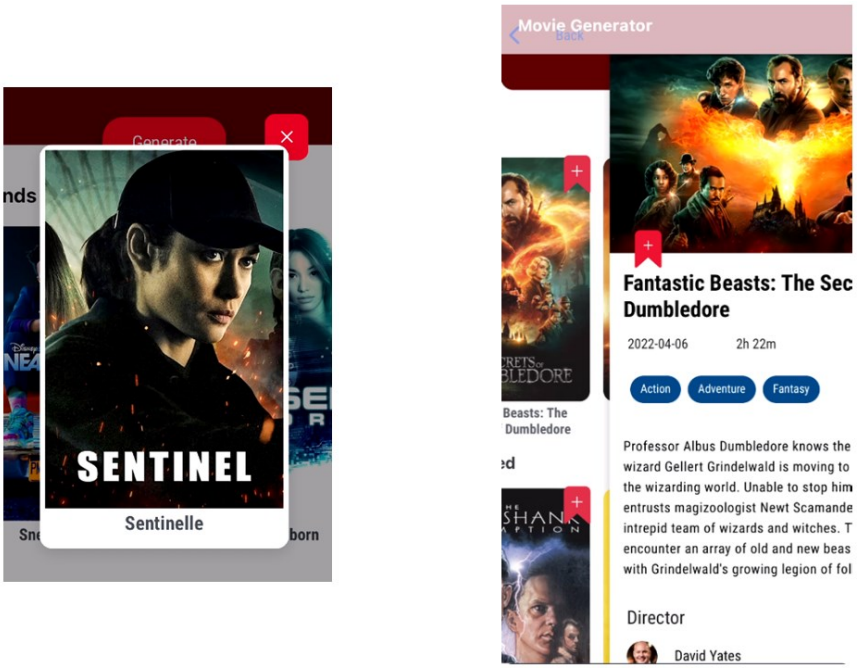
Stránka s filmem, přidávání nebo odebírání položek z lokální paměti funguje bez problémů.

8.2 Android aplikace

Zkoušel jsem testovat android aplikaci v Android studiu, ale po načtení emulátoru a spuštění aplikace byl problém s výkonem celého emulátoru, a hlavně začalo hodně věcí špatně fungovat, obrázky se nenačítaly celé a vše bylo takové zpomalené. Kvůli tomu jsem sáhl po reálném zařízení, které bylo pomocí USB ladění připojeno k Android studiu a aplikace tak testována přes tento mobil. Po vyzkoušení celé funkčnosti aplikace vypadá, že funguje správně i s přechody mezi stránkami, jediné, co je možné vypozorovat je delší prodleva načítání různých položek, ale to bude zapříčiněno starším modelem telefonu.

8.3 PWA

Zbývá poslední platforma a tou je iOS, na kterém se testování provádělo za pomocí progresivní webové aplikace, kterou lze z prohlížeče přímo nainstalovat do zařízení iPhone. Tady došlo na pár věcí, které se chovaly jinak, než měly, protože iPhone obsahuje navíc ovládání pomocí gest, tak je možné se vrátit ze stránky filmu na domovskou stránku swipnutím zleva doprava a v hodně případech ta animace přechodu zpět není úplně čistá a objeví se tam krátký zásek. Dalším problémem, který vzniknul je při načtení modal okna, když se vygeneruje náhodný film tak se křížek u pár případů nachází až za samotným obrázkem a je tak zakrytý.



Obrázek 8.2 Problém se zobrazením tlačítka na zavření (vlevo) a problém s animací při přechodu zpět (vpravo)

Posledním problémem je nemožnost skrolovat v aplikaci na domovské stránce po náhodné kombinaci klikání na generování a vrácení se zpět. Tento problém však nevznikal tak často. Jinak nedošlo na žádné další kritické problémy a výkon celé aplikace byl na dobré úrovni.

9 PRAKTICKÉ POROVNÁNÍ JS FRAMEWORKŮ

Tato část se věnuje porovnávání JS frameworků z hlediska předem stanovených faktorů, které ovlivňují vývoj celé aplikace nebo webové stránky.

9.1 Faktory ovlivňující volbu JS frameworku

V této kapitole se popisují odlišnosti podle předem zvolených faktorů.

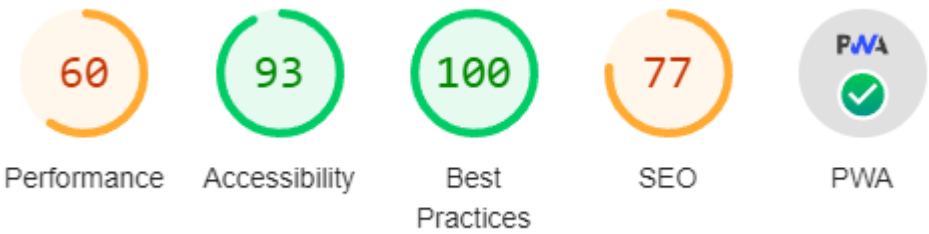
9.1.1 Rychlost

První porovnání proběhlo v Android studiu, kde se testovalo za jak dlouho je schopna se aplikace načíst, přesněji ne úplné načtení, ale jakmile se začne něco dít, stačí i splash screen.

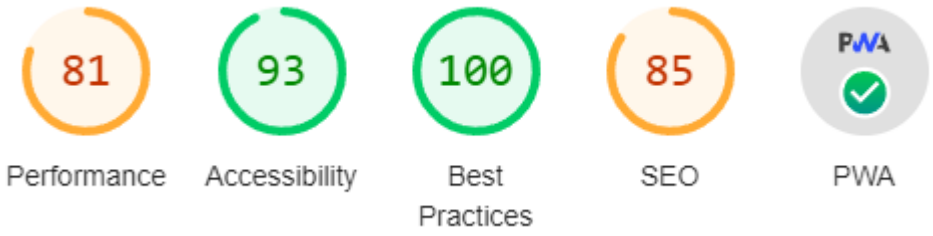
Tabulka 9.1 Průměrná doba trvání načtení aplikace

Samsung Galaxy S8	Angular	React	Vue
Průměrný čas	0,903 sekund	0,8317 sekund	0,8334 sekund

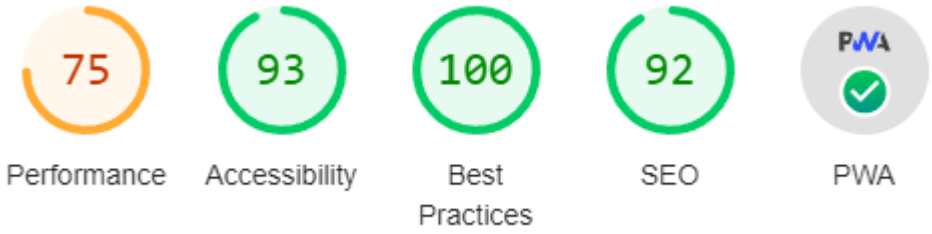
Aplikace také byla podrobena celkovému testu v prohlížeči zvanému Lighthouse, který má za úkol probrat ty nejvíc klíčové aspekty týkající se webové stránky nebo aplikace, tím je výkon, který se určuje podle rychlosti načtení, podle velikosti obsahu, co je potřeba zobrazit. Přístupnost ukazuje metriku, na jaké úrovni se dá zpřístupnit daná aplikace lidem s nějakým handicapem. Dalším prvkem je dodržení určitých pravidel dané platformy např. tlačítko na přidání by mělo být v pravém dolním rohu v mobilní aplikaci. SEO udává, jak moc je aplikace nebo web optimalizována pro vyhledávače a poslední metrika je pouze v případě, pokud se jedná o PWA.



Obrázek 9.1 Angular Lighthouse



Obrázek 9.2 React Lighthouse



Obrázek 9.3 Vue Lighthouse

9.1.2 Velikost projektu

Tato část se bude věnovat velikosti projektu, to znamená počet souborů a složek a následná orientace v celé struktuře.

Nejdříve se podíváme na **Angular**, od kterého se předpokládá, že zabere nejvíc místa, protože na rozdíl od Reactu to je plnohodnotný framework a Vue se dá považovat jako odlehčená verze. Co se týká struktury projektu tak obsáhlost souborů a složek je hodně a může se stát, že programátor se začne trochu ztrácet v orientaci. Každopádně tomu může dopomáhat rozdělená komponenta do několika souborů, kdy všechno není na jednom místě a každý soubor má svoje opodstatnění.

Na **Reactu** lze vypozorovat, že jeho struktura neobsahuje tolik složek a souborů, za to se může zde nacházet více malých komponent, ale její obsah může být napsaný do jediného souboru, ve kterém se sice může špatně orientovat, ale na venek to nevypadá tak hrozně složitě.

Vue podobně jako React má lepší orientaci ve struktuře celého projektu, za to uvnitř souboru komponent se to může zdát skutečně přehlednější, protože obsah je rozdělený do 3 bloků a každý z nich slouží pro jiný účel.

9.1.3 Množství vloženého obsahu (knihoven)

Zde se bude rozebírat kolik místa zabírá projekt na disku a množství importovaného obsahu pro různé potřeby aplikace.

Angular aplikace, která byla tvořena zabírá na disku „pouhých“ 9,60 MB bez node balíčků a 574 MB s nimi. Množství vloženého obsahu je zde minimum, protože hodně toho už Angular má v sobě a není tak potřeba něco navíc doinstalovávat, jediné, co bylo potřeba je samotný ionic a přidat balíček na vytvoření PWA.

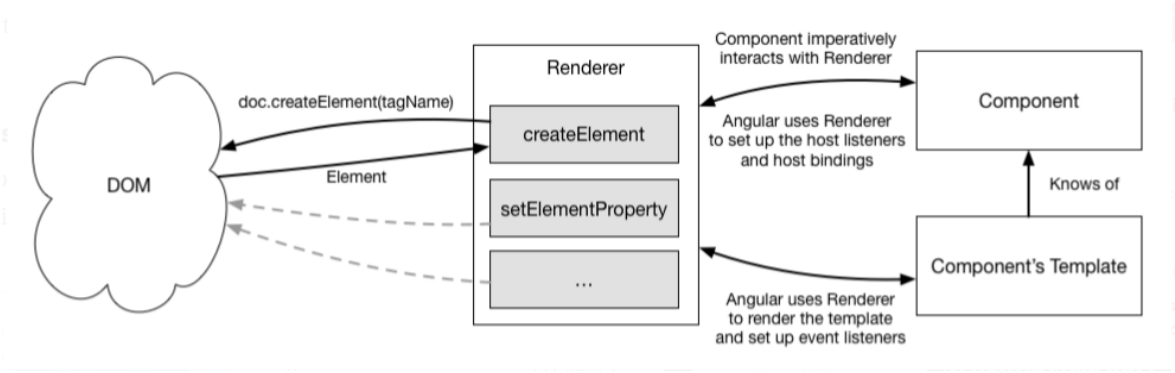
U **Reactu** se velikost pohybuje okolo 5,51 MB bez node balíčků a 386 MB s nimi, a tak obsahu, co bylo potřeba dodatečně doinstalovat bylo hodně a ten rozdíl je veliký. Množství importovaného obsahu zabírá několik řádků skoro v každé komponentě a mimo ionic nebo PWA tu bylo potřeba instalovat routery a další knihovny.

Aplikace ve **Vue** dosahuje velikosti 5,43 MB bez node balíčků a 344 MB s nimi, tím se Vue řadí mezi nejmenší a jeho odlehčenost se tu potvrzuje. Vložený obsah se pohybuje na minimu a připomíná tak Angular až na to, že je potřeba stejně jako v Reactu importovat každou ionic komponentu zvlášť a nenabízí importování přímo celého modulu.

9.1.4 Vytváření a renderování komponent za využití props

Samotné vytváření komponent probíhalo odlišně, kdy se konkrétně u **Angularu** komponenta generovala pomocí příkazu a člověk tak nemusel vytvářet jednotlivé soubory a pamatovat si co všechno musí obsahovat. Navíc příkaz dokáže už předem definovat komunikaci mezi soubory, tak aby komponenta fungovala bez problému, tím se myslí, že například označí název pro použití komponenty a propojí HTML šablonu a CSS styly a taky připraví základní strukturu programové logiky, jako je třída, konstruktor, případně podle potřeby životní cyklus. Proměnné „props“ se do třídy zapisovaly pomocí klíčového slova @Input(), znázorňující data, které slouží jako vstupní parametry dané komponenty, do kterých jde přiřadit libovolná hodnota podle zvoleného datového typu. Pokud je však potřeba proměnná jako výstup existuje možnost zapsání @Output(), ovšem tento způsob se používá nejčastěji na převod celé funkce jako „props“ mezi rodičem a dítětem.

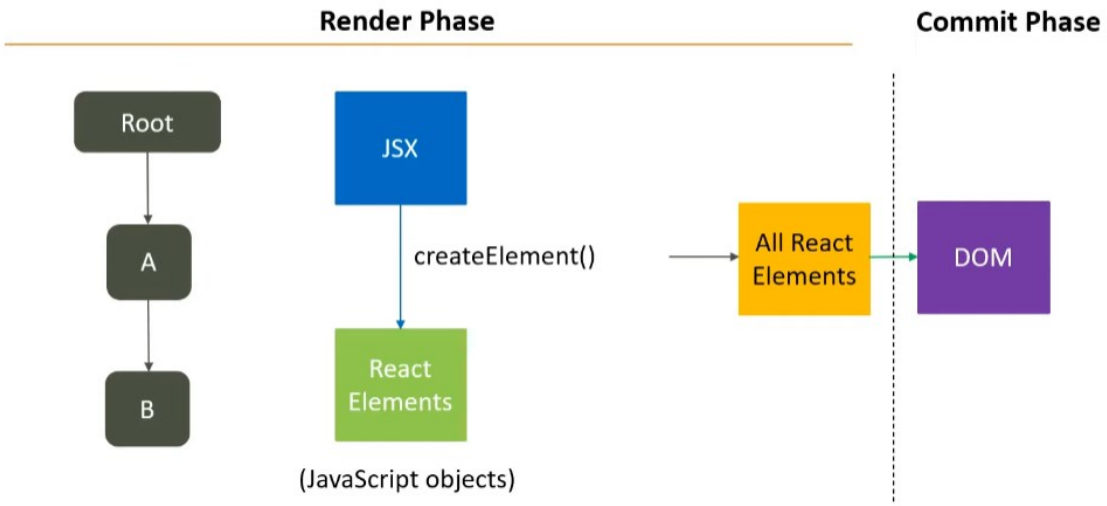
Angular renderuje komponenty způsobem, kdy komponenta ví o šabloně a snaží se interaktivně reagovat na její změny, aby se nastavily správné elementy, které se budou vytvářet. V renderování se potom tato komponenta vytvoří jako klasicky HTML tag, který se zobrazuje ve výsledném DOM společně s jejím obsahem [41].



Obrázek 9.4 Princip renderování Angularu [41]

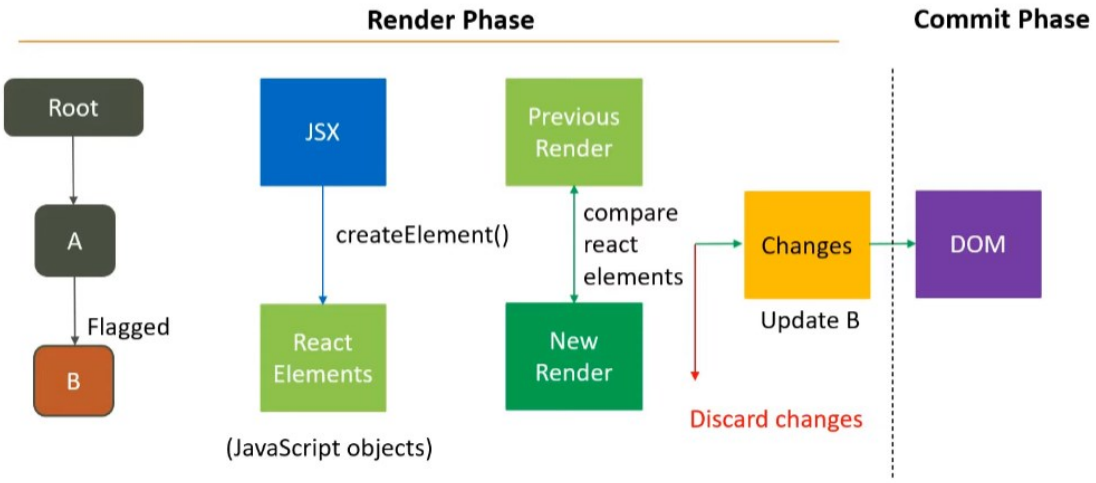
U **Reactu** se komponenty tvoří manuálním způsobem, tedy vytvoří se soubor s koncovkou .jsx nebo .tsx podle toho, který jazyk se zvolí a do něho se potom píše celý obsah. Rozdílem je, že komponenta představuje jeden soubor, ve kterém se nachází programová logika i to co se má renderovat do výsledné aplikace, jediné, co dává smysl oddělit do dalšího souboru jsou CSS styly. Jestli je potřeba „props“ tak ty se definují přímo v parametru funkce uvnitř komponenty anebo je možné si pro ně udělat zvláštní rozhraní, kde se zapisují společně s datovým typem a rozhraní je potom předáno funkci.

Renderování komponent v Reactu se rozděluje na dvě fáze, renderovací fáze a fáze, kde je výsledek renderování odevzdán DOMu. V renderovací fázi se nejdřív postupně prochází od root komponenty (Root) až po navazující (A a B), při procházení se u každé komponenty zavolá metoda createElement(), která všechno převede na React elementy, které se potom aplikují do DOMu [42].



Obrázek 9.5 Počáteční vykreslení Reactu [42]

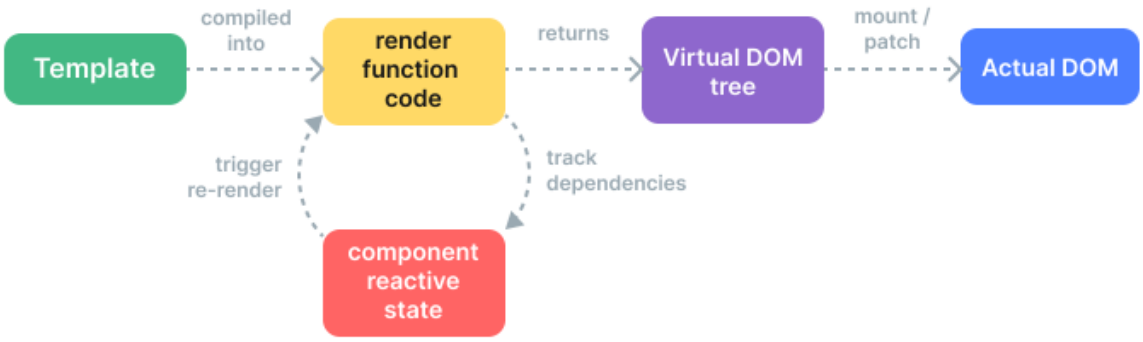
Toto bylo popsání renderu při první inicializaci, ale pokud se provede změna tak je potřeba tuto změnu vykreslit znovu. Průběh bude podobný až na to, že se zkontroluje pouze komponenta, v niž bylo něco upraveno a po vytvoření React elementů se tento výstup porovná s předchozím výstupem a změny se aplikují do DOMu. Důležité je si všimnout, že pokud se žádné úpravy neprovedou, tak se elementy vyhodí [42].



Obrázek 9.6 Opakované vykreslení reagující na změnu [42]

Vytváření komponent ve **Vue** funguje na podobném principu jako React s tím rozdílem, že se koncovka souboru nazývá .vue a „props“ jsou definované ve script tagu, přímo v komponentě. CSS styly se nachází taky v tomto souboru a je tak vlastně všechno na jednom místě.

Co se týče renderování probíhá tak, že je obsah HTML šablony převeden do funkcí, které sledují závislosti stavů jednotlivých komponent a případné změny zapříčiní jejich opětovné vykreslení. Jakmile je všechno zkontrolováno, tak se vrátí nový virtuální DOM a porovná se s tím starým a potřebné úpravy převede do klasického DOMu [43]



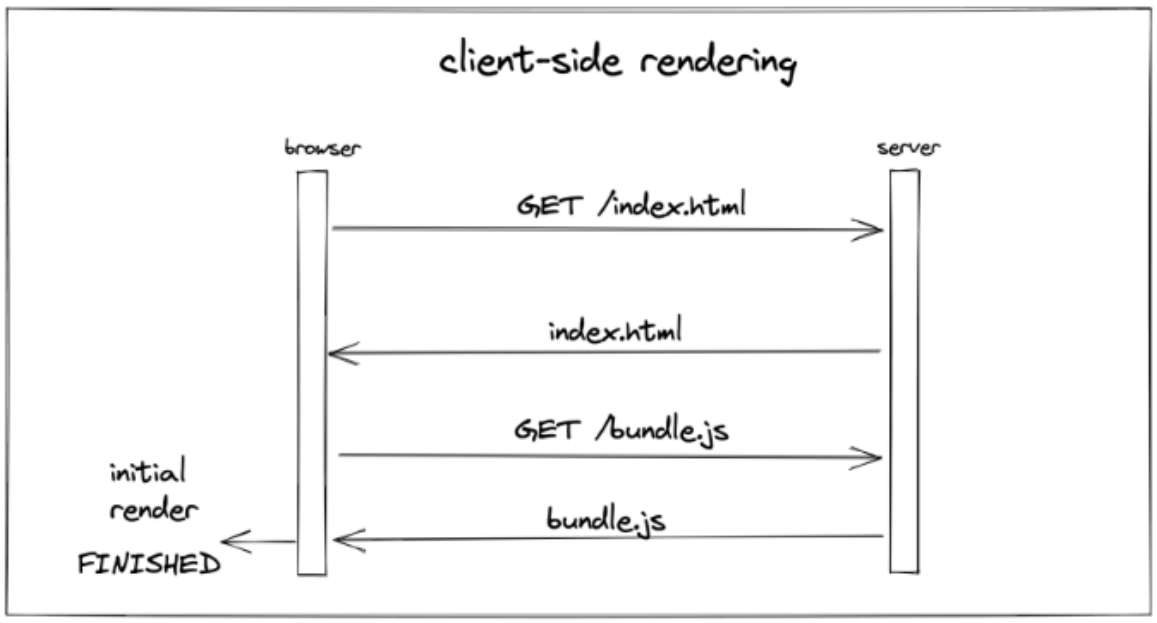
Obrázek 9.7 Princip renderování Vue [43]

9.1.5 Client-side a Server-side rendering

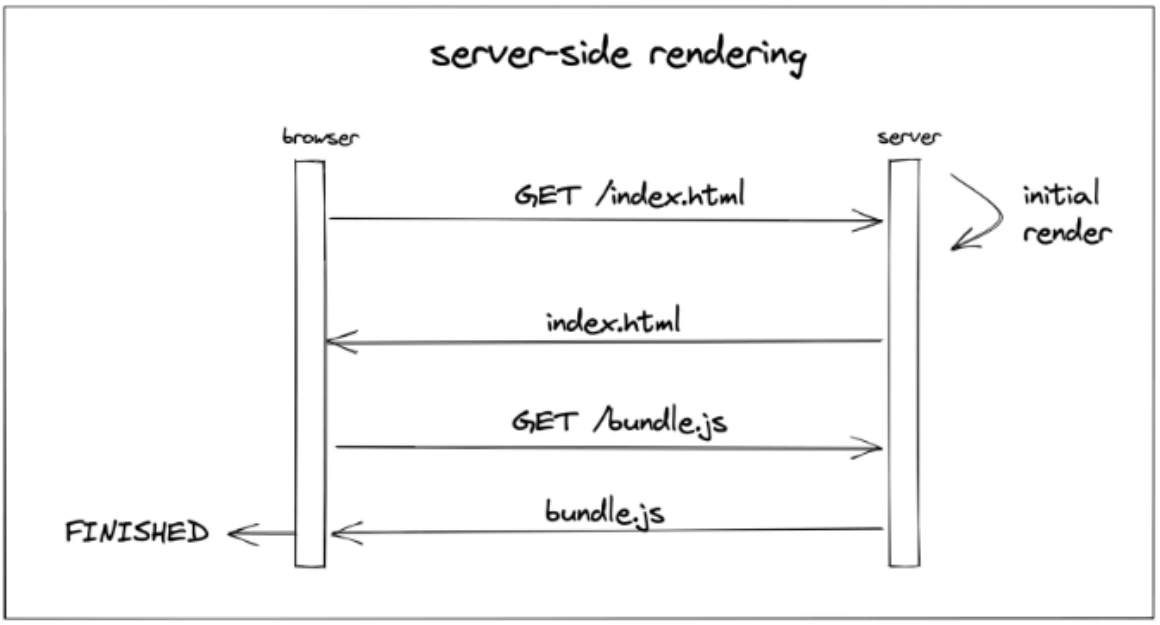
Probírané JS frameworky jsou všechny zpracovány tak, že se jejich kód renderuje na straně klienta a je potřeba dalších nástrojů, které danou aplikaci předělají na server.

Na vytvoření aplikace v **Angularu**, která bude běžet na serveru, je potřeba speciální knihovny s názvem Angular Universal, pomocí ní můžeme vykreslovat aplikaci předem, tedy uživatel uvidí obsah, hlavně HTML a CSS na stránce při prvním požadavku. S tímto vykreslením se dodává taky běžná aplikace na straně klienta, která pak vezme kontrolu nad vším a chová se jak client-side [44].

React nabízí možnost vytvořit server-side aplikaci pomocí frameworků, které běží jak na straně klienta, tak i na serveru, mezi tyto frameworky patří Next.js, Remix.run atd. Vykreslování na straně klienta probíhá tak, že si prohlížeč vyžádá soubor index.html, nejčastěji v public složce, ve kterém jsou definované CSS styly a další obsah, který se má načíst. Tento obsah se načte pouze tehdy, až se dokončí stažení celého JavaScriptu a následně bude moct být spuštěn, zatímco vykreslování na straně serveru probíhá tak, že se importuje kořenová aplikace, do které je všechen obsah renderovaný (ReactDOM.render) a ta se vykreslí do dokumentu, který se následně vrátí klientovi, zjednodušeně řečeno se nejdřív zobrazí počáteční obsah HTML na serveru a nemusí se tak čekat, než se načte JavaScript v prohlížeči [45].



Obrázek 9.8 Renderování na straně klienta [45]



Obrázek 9.9 Renderování na straně serveru [45]

Vue aplikace na straně serveru funguje na stejném principu, komponenty se dají vykreslit, konkrétně jejich HTML obsah a tento obsah potom následně poslat do prohlížeče, kde se po chvilce aplikace převede na stranu klienta. Takto vykreslená aplikace uchovává velké množství kódu na obou stranách a hodně funkcí je tu v počátku vypnutých, protože se na serveru neprovádí žádné interakce s uživatelem a DOM se nemění. Pro náročnější projekty připadají v úvahu použití frameworku Nuxt, který je založen na stejném ekosystému jako Vue [46].

ZÁVĚR

Cílem bakalářské práce bylo seznámení čtenáře s JS frameworky, jejich odlišnostmi v celkovém postupu vývoje a použitelnosti na tvorbu specifické aplikace nebo webu.

Teoretická část představila přehled o známých typech aplikací, které lze vyvíjet a sestavila tak základní pojetí o tom co jaký typ znamená a jaké technologie je potřeba umět na jejich vytvoření. Potom byl popsán hlavní programovací jazyk práce JavaScript, konkrétně jeho velká využitelnost napříč celým vývojem softwaru a porovnání jazyka jako takového s TypeScriptem. Následovala hlavní kapitola bakalářské práce, kde se představuje čtenářovi úvod do světa JS frameworků a jejich ulehčení vývoje oproti použití obyčejného JavaScriptu. Nachází se zde vybrané frameworky, které jsou následně popsány, na jakém principu fungují a co všechno obsahují za funkce.

Praktická část popsala úplný vývoj celé aplikace, konkrétně jakých nástrojů a programů bylo potřeba na samotný vývoj a potom jakým způsobem se vytváří projekt až po kompletní zpracování aplikace. Následně se muselo provést testování, zda všechno funguje bez problémů na všech různých platformách a případné problémy zmínit. Důležitou součástí bylo představení faktorů, podle kterých si čtenář dokáže odvodit, jaký JS framework je pro specifický projekt vhodný, probíhalo zde porovnání, kdy byly aplikace testovány určitými nástroji a výsledky byly vhodně prezentovány.

V této práci by mohli najít přínos hlavně programátoři, kteří mají už nějaké znalosti s JavaScriptem a rozhodují se nyní jaký framework by si měli vybrat pro další posun v jejich cestě učení se programování, ale taky pro jaký specifický projekt se bude více hodit. Popis vývoje aplikace totiž slouží jako takový návod, jak vůbec s frameworky začít vyvíjet nějakou aplikaci. Zálibu ve čtení by mohli najít i lidé, kteří už s danými technologiemi nějakou dobu pracují a potřebují rozšířit své znalosti o tom na jakém principu fungují a co přesně probíhá při vývoji.

SEZNAM POUŽITÉ LITERATURY

[1] Webová, nativní a hybridní aplikace: srovnáváme pro a proti. *Rascasone* [online]. Praha: Rascasone, 2021 [cit. 2022-05-16]. Dostupné z: https://www.rascasone.com/cs/blog/webova-nativni-hybridni-aplikace-klady-zapory

[2] JavaScript. *MDN Web Docs* [online]. San Francisco: Mozilla Corporation, c1998–2022 [cit. 2022-05-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/web/javascript

[3] JavaScript basics - Learn web development. *MDN Web Docs* [online]. San Francisco: Mozilla Corporation, c1998–2022 [cit. 2022-05-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[4] Introduction to the DOM - Web APIs. *MDN Web Docs* [online]. San Francisco: Mozilla Corporation, c1998–2022 [cit. 2022-05-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

[5] TypeScript: Documentation - TypeScript for the New Programmer. *TypeScript: JavaScript With Syntax For Types.* [online]. Redmond: Microsoft Corporation, c2012-2022 [cit. 2022-05-16]. Dostupné z: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html

[6] What Is JavaScript Framework. *General Assembly* [online]. Urban, c2022 [cit. 2022-05-17]. Dostupné z: https://generalassemb.ly/blog/what-is-a-javascript-framework/

[7] Stack Overflow Developer Survey 2021. *Stack Overflow - Where Developers Learn, Share, & Build Careers* [online]. 2021 [cit. 2022-05-21]. Dostupné z: https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks

[8] The Model View Controller Pattern – MVC Architecture and Frameworks Explained. *FreeCodeCamp Programming Tutorials: Python, JavaScript, Git & More* [online]. San Francisco: Quincy Larson, 2021 [cit. 2022-05-16]. Dostupné z: https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained

[9] *What is a JavaScript Framework? (in detail)* [online]. Youtube, 2020 [cit. 2022-05-17]. Dostupné z: https://www.youtube.com/watch?v=Ka77djMkSwg

[10] Proč zvolit právě technologii React?. *Rascasone* [online]. Michaela Kadlecová, 2022 [cit. 2022-05-19]. Dostupné z: https://www.rascasone.com/cs/blog/react-js-pro-svizne-a-moderni-weby-a-aplikace

[11] React – A JavaScript library for building user interfaces. *React – A JavaScript library for building user interfaces* [online]. Kalifornie: Meta Platforms, c2022 [cit. 2022-05-18]. Dostupné z: https://reactjs.org/

[12] Rendering Elements. *React – A JavaScript library for building user interfaces* [online]. Kalifornie: Meta Platforms, c2022 [cit. 2022-05-18]. Dostupné z: https://reactjs.org/docs/rendering-elements.html

[13] Introducing JSX. *React – A JavaScript library for building user interfaces* [online]. Kalifornie: Meta Platforms, c2022 [cit. 2022-05-18]. Dostupné z: https://reactjs.org/docs/introducing-jsx.html#jsx-represents-objects

[14] Components and Props. *React – A JavaScript library for building user interfaces* [online]. Kalifornie: Meta Platforms, c2022 [cit. 2022-05-18]. Dostupné z: https://reactjs.org/docs/components-and-props.html

[15] Introducing Hooks. *React – A JavaScript library for building user interfaces* [online]. Kalifornie: Meta Platforms, c2022 [cit. 2022-05-18]. Dostupné z: https://reactjs.org/docs/hooks-intro.html

[16] Using the Effect Hook. *React – A JavaScript library for building user interfaces* [online]. Kalifornie: Meta Platforms, c2022 [cit. 2022-05-18]. Dostupné z: https://reactjs.org/docs/hooks-effect.html

[17] Using the State Hook. *React – A JavaScript library for building user interfaces* [online]. Kalifornie: Meta Platforms, c2022 [cit. 2022-05-19]. Dostupné z: https://reactjs.org/docs/hooks-state.html

[18] What is Angular?. *Angular* [online]. Kalifornie: Google, c2010-2022 [cit. 2022-05-19]. Dostupné z: https://angular.io/guide/what-is-angular

[19] Co je Angular, v čem je jiný než AngularJS a proč ho použít?. *Rascasone* [online]. Barbora Koďousková, 2021 [cit. 2022-05-19]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-angular-angularjs#co-je-angular

[20] Angular Components Overview. *Angular* [online]. Kalifornie: Google, c2010-2022 [cit. 2022-05-19]. Dostupné z: https://angular.io/guide/component-overview

[21] Lifecycle hooks. *Angular* [online]. Kalifornie: Google, c2010-2022 [cit. 2022-05-19]. Dostupné z: https://angular.io/guide/lifecycle-hooks

[22] Built-in directives. *Angular* [online]. Kalifornie: Google, c2010-2022 [cit. 2022-05-19]. Dostupné z: https://angular.io/guide/built-in-directives

[23] Dependency injection in Angular. *Angular* [online]. Kalifornie: Google, c2010-2022 [cit. 2022-05-19]. Dostupné z: https://angular.io/guide/dependency-injection

[24] Introduction. *Vue.js - The Progressive JavaScript Framework* [online]. Evan You, c2014-2022 [cit. 2022-05-19]. Dostupné z: https://vuejs.org/guide/introduction.html

[25] Vue.js: tvorba svižných webů a vývoj single-page aplikací. *Rascasone* [online]. Vít Uličný, 2021 [cit. 2022-05-19]. Dostupné z: https://www.rascasone.com/cs/blog/vue-js-lehky-framework-idealni-pro-tvorbu-webovych-komponent

[26] Components Basics. *Vue.js - The Progressive JavaScript Framework* [online]. Evan You, c2014-2022 [cit. 2022-05-19]. Dostupné z: https://vuejs.org/guide/essentials/component-basics.html

[27] Reactivity in Depth. *Vue.js - The Progressive JavaScript Framework* [online]. Evan You, c2014-2022 [cit. 2022-05-19]. Dostupné z: https://vuejs.org/guide/extras/reactivity-in-depth.html

[28] Lifecycle Hooks. *Vue.js - The Progressive JavaScript Framework* [online]. Evan You, c2014-2022 [cit. 2022-05-19]. Dostupné z: https://vuejs.org/guide/essentials/lifecycle.html

[29] Guide to Ionic framework V4+: Which front-end framework to use. *Educative: Interactive Courses for Software Developers* [online]. c2022 [cit. 2022-05-19]. Dostupné z: https://www.educative.io/blog/ionic-framework-v4p

[30] The Ionic Platform. *Ionic Docs* [online]. Wisconsin: Drifty Co, c2022 [cit. 2022-05-19]. Dostupné z: https://ionic.io/docs/platform

[31] Cross-Platform Mobile App Development: Ionic Framework. *Cross-Platform Mobile App Development: Ionic Framework* [online]. Wisconsin: Drifty Co, c2022 [cit. 2022-05-19]. Dostupné z: https://ionicframework.com/

[32] Introduction to Ionic. *Ionic Framework* [online]. Wisconsin: Drifty Co, c2022 [cit. 2022-05-19]. Dostupné z: https://ionicframework.com/docs

[33] What the Heck is Shadow DOM?. *Dimitri Glazkov* [online]. Dimitri Glazkov, 2011 [cit. 2022-05-19]. Dostupné z: https://glazkov.com/2011/01/14/what-the-heck-is-shadow-dom/

[34] Capacitor - Cross-platform Native Runtime for Web Apps - Capacitor. *Capacitor by Ionic - Cross-platform apps with web technology* [online]. Wisconsin: Drifty Co, c2022 [cit. 2022-05-19]. Dostupné z: https://capacitorjs.com/docs

[35] Capacitor vs Cordova: Hybrid Mobile App Development. *Ionic: Enterprise App Development & Delivery Platform* [online]. Wisconsin: Drifty Co, c2022 [cit. 2022-05-19]. Dostupné z: https://ionic.io/resources/articles/capacitor-vs-cordova-modern-hybrid-app-development

[36] Angular Architecture Explained. *Edureka* [online]. Brain4ce Education Solutions, c2022 [cit. 2022-05-19]. Dostupné z: https://www.edureka.co/blog/angular-mvc-architecture/

[37] Vue vs. Angular: Which Framework to Choose in 2022? - Trio Developers. *The Trio Blog* [online]. Cordenne Brewster, c2022 [cit. 2022-05-19]. Dostupné z: https://www.trio.dev/blog/vue-vs-angular

[38] When should I use curly braces { } and parenthesis ( ) in React?. *JavaScript in Plain English* [online]. Leanne Zhang, 2018 [cit. 2022-05-19]. Dostupné z: https://javascript.plainenglish.io/curly-braces-versus-parenthesis-in-reactjs-4d3ffd33128f

[39] Property binding. *Angular* [online]. Kalifornie: Google, c2010-2022 [cit. 2022-05-19]. Dostupné z: https://angular.io/guide/property-binding

[40] Template Syntax. *Vue.js* [online]. Evan You, c2014-2022 [cit. 2022-05-19]. Dostupné z: https://vuejs.org/guide/essentials/template-syntax.html

[41] Experiments with Angular Renderers. *Medium* [online]. Victor Savkin, 2016 [cit. 2022-05-19]. Dostupné z: https://blog.nrwl.io/experiments-with-angular-renderers-c5f647d4fd9e

[42] *React Render Tutorial* [online]. Kalifornie: Codevolution, 2020 [cit. 2022-05-19]. Dostupné z: https://www.youtube.com/watch?v=VPtL6dU0YXc

[43] Rendering Mechanism. *Vue.js* [online]. Evan You, c2014-2022 [cit. 2022-05-19]. Dostupné z: https://vuejs.org/guide/extras/rendering-mechanism.html

[44] Angular Universal: Complete Practical Guide. *Angular University* [online]. Angular University, 2022 [cit. 2022-05-19]. Dostupné z: https://blog.angular-university.io/angular-universal/

[45] Basics of React server-side rendering with Express.js. *DEV Community* [online]. Juhana Jauhiainen, 2022 [cit. 2022-05-19]. Dostupné z: https://dev.to/juhanakristian/basics-of-react-server-side-rendering-with-expressjs-phd

[46] Server-Side Rendering (SSR). *Vue.js* [online]. Evan You, c2014-2022 [cit. 2022-05-19]. Dostupné z: https://vuejs.org/guide/scaling-up/ssr.html

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UI	User Interface
UX	User Experience
SPA	Single-page application
PWA	Progressive Web Apps
API	Application Programming Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
DOM	Document Object Model
XML	Extensible Markup Language
OOP	Object-oriented programming
MVC	Model-view-controller
MVVM	Model-view-viewmodel
JSON	JavaScript Object Notation
SDK	Software development kit
NPM	Node Package Manager
CLI	Command-line interface
SASS	Syntactically Awesome Style Sheets

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: NÁZEV PŘÍLOHY

fulltext.pdf	Textová část bakalářské práce
Angular/	Zdrojové kódy Angular aplikace
React/	Zdrojové kódy React aplikace
Vue/	Zdrojové kódy Vue aplikace