VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná Informatika

POROVNÁNÍ NÁSTROJŮ PRO VÝVOJ OPTIMALIZOVANÝCH WEBOVÝCH STRÁNEK A APLIKACÍ

Bakalářská práce

Autor práce: Matěj Košín

Vedoucí práce: PaedDr. František Smrčka, Ph.D.

Jihlava 2024

Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce:	**Matěj Košín**
Studijní program:	Aplikovaná informatika
Obor:	Aplikovaná informatika
Garant studijního programu:	Ing. Lenka Kuklišová Pavelková, Ph.D.
Název práce:	**Porovnání nástrojů pro vývoj optimalizovaných webových stránek a aplikací**
Vedoucí práce:	PaedDr. František Smrčka, Ph.D.
Cíl práce:	Cílem práce je popsat a vzájemně porovnat nástroje pro vývoj moderních a optimalizovaných webových stránek a aplikací, tj. JavaScriptové frameworky a knihovnu React. V rámci práce bude pro každou z porovnávaných technologií vyvinuta jednoduchá aplikace a tyto budou následné porovnány z hlediska vývojářské přívětivosti.

Abstrakt

Bakalářská práce se věnuje analýze a porovnání současných frontendových frameworků a knihovny React, které jsou klíčovými nástroji v oblasti vývoje moderních a optimalizovaných webů a webových aplikací. Hlavním cílem práce je poskytnout ucelený pohled na nejpoužívanější JavaScriptové frameworky – specificky Vue.js, React a Astro – a identifikovat jejich klíčové vlastnosti, silné stránky a omezení. Práce zahrnuje teoretickou část, ve které jsou tyto technologie představeny a porovnány s ohledem na jejich architekturu, výkon, škálovatelnost a vývojářskou přívětivost. Důležitou součástí jsou také kaskádové styly, které také mají různé frameworky. V bakalářské práci se zaměříme na framework Tailwind.

V praktické části je pro dvě vybrané technologie vyvinuta jednoduchá aplikace, díky čemuž je možné porovnat reálné zkušenosti z vývoje a poukázat na specifika každého frameworku a knihovny. Toto porovnání je klíčové pro pochopení, jak rozdílné přístupy ovlivňují proces vývoje a jaké výhody a nevýhody mohou mít pro konkrétní typy projektů.

Klíčová slova

Front-end Framework; React; Vue.js; Astro; Javascript; Typescript; Komponenta;

Abstract

This bachelor thesis is dedicated to the analysis and comparison of current frontend frameworks and the React library, which are key tools in the development of modern and optimized websites and web applications. The main goal of the thesis is to provide a comprehensive view of the most widely used JavaScript frameworks - specifically Vue.js, React and Astro - and identify their key features, strengths and limitations. The thesis includes a theoretical section in which these technologies are introduced and compared with respect to their architecture, performance, scalability and developer friendliness. Cascading stylesheets, which also have different frameworks, are also an important part. In this paper, we focus on the Tailwind framework.

In the practical part, a simple application is developed for the two selected technologies, which allows us to compare real development experiences and highlight the specifics of each framework and library. This comparison is crucial to understand how different approaches affect the development process and what advantages and disadvantages they may have for specific types of projects.

Keywords

Front-end Framework; React; Vue.js; Astro; Javascript; TypeScript; Component;

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „**AZ**“).

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Podle § 47b zákona o vysokých školách souhlasím se zveřejněním své práce podle Směrnice pro vedení, vypracování a zveřejňování závěrečných prací na VŠPJ, a to bez ohledu na výsledek obhajoby.

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mě požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Třebíči dne 20. listopadu 2024

…….

Podpis studenta

Poděkování

Rád bych poděkoval vedoucímu práce, panu PaedDr. Františku Smrčkovi Ph.D., který mi byl k dispozici při mých dotazech. Dále paní Mgr. Haně Vojáčkové, Ph.D., za její ochotu a pomoc při studiu. Velký dík patří také mým spolužákům, kteří mě během studia několikrát podpořili, a neposlední řadě mé rodině, která se mnou měla po celou dobu studia trpělivost.

Obsah

Seznam obrázků

Seznam Tabulek

Seznam zkratek

CSS	Cascading style sheets (kaskádové styly)
DHTML	Dynamic HTML
DOM	Document Object Model
HTML	HyperText Markup Language (hypertextový značkovací jazyk)
JIT	Just-In-Time
JSX	JavaScript XML
VŠPJ	Vysoká škola polytechnická Jihlava
XML	Extensible Markup Language
TSX	TypeScript XML
UI	User interface

Úvod

Když se řekne pojem frontend, mnohým se vybaví základní HTML a CSS, které tvoří kostru webových stránek. Avšak v dnešní době tento termín nabývá mnohem širšího významu. Moderní frontend vývoj již dávno přesáhl své tradiční hranice a proměnil se v dynamickou disciplínu, která v sobě integruje nejen estetické aspekty, ale také rozsáhlé programovací schopnosti. V dnešním digitálním světě, kde kvalita a rychlost webových aplikací jsou nezbytné pro udržení pozornosti uživatelů, se frontendový vývoj stává klíčovým prvkem v procesu vytváření uživatelsky přívětivých a technologicky pokročilých webových řešení.

Existuje široká škála JavaScriptových frameworků a knihoven, jako jsou React, Vue.js, Angular, a mnoho dalších, které umožňují vývojářům tvořit interaktivní a vizuálně atraktivní weby a webové aplikace. Tato nástrojová sada přináší nejen estetické výhody, ale také zásadně zlepšuje uživatelskou zkušenost na webu. Důležitým aspektem těchto technologií je jejich vývojářská přívětivost, což znamená, že nabízejí efektivní a intuitivní prostředky pro rychlý a plynulý vývoj.

Výběr správného nástroje pro konkrétní projekt však může být pro vývojáře výzvou. Každý framework a knihovna má své unikátní vlastnosti, silné stránky a omezení, a není jednoduché určit, který z nich je „nejlepší“. V mé bakalářské práci se proto zaměřuji na základní porovnání několika nejpopulárnějších frontendových technologií. Cílem je poskytnout přehledný a objektivní pohled na jejich funkce, výhody a potenciální slabiny.

Porovnání nebude jen teoretické. Pro každý vybraný framework nebo knihovnu vyvinu jednoduchou aplikaci, což mi umožní poskytnout praktické informace založené na reálných zkušenostech. Věřím, že moje práce pomůže začínajícím vývojářům lépe porozumět rozmanitosti dostupných nástrojů a usnadní jim výběr té správné technologie pro jejich budoucí projekty.

Proč zrovna téma frontendových frameworků? V prvním ročníku VŠPJ jsem si zamiloval HTML/CSS a došlo mi, že by mě bavilo pracovat jako frontendový vývojář. Nemilého zjištění, že pouze HTML/CSS už v dnešní době zkrátka nestačí, jsem se nezalekl a ponořil se do studia moderních technologií. Velmi se mi v oblasti vývoje webů líbí, že vývoj jde neustále dopředu a je motivace se stále učit.

1 Teoretická část

Kapitola slouží pro základní uvedení čtenáře do teorie problematiky frontendového vývoje a frameworků pro něho využívaných. Popisuje mimo jiné i rozdíly mezi frameworky i programovacími jazyky JavaScript a TypeScript.

1.1 Přehled frontendových technologií

Frontendové technologie čítají nepřeberné množství možností, které lze použít pro vývoj. Od toho slouží následujících pár odstavců, kde jsou některé z těchto technologií vysvětleny.

1.1.1 Co je to frontend

Frontend, v doslovném překladu přední část, je, jak již název napovídá, to, co návštěvník webové stránky či aplikace vidí, nebo jak s ní může interagovat. Jedná se o design uživatelského rozhraní, uživatelské zkušenosti a také všechny části stránky, které může návštěvník jakýmkoliv způsobem ovlivnit, např.: formuláře, jež mohou být vyplněny a odeslány.

Nezbytností webové stránky je HTML, značkovací hypertextový jazyk. Ten definuje obsah a jeho strukturu. Dále pak kaskádové styly, jejichž zkratka je CSS, které se používají pro vizuální formátování a design stránek, a to včetně rozložení stránek, barev, fontů a dalších vizuálních prvků. Samotnou interaktivitu a dynamičnost webu pak obstarává programovací jazyk JavaScript. S jeho pomocí lze implementovat složité funkce, jako jsou animace, načítání dat, manipulace s prvky HTML, upravování CSS a mnoho dalšího.

Celé se to však týká také otázky přístupnosti (aby web byl přístupný také lidem s různými zdravotními postiženími), responzivity (správné zobrazení webu na zařízeních všech typů a velikostí obrazovek) a optimalizace výkonu (rychlé načítání obsahu stránek).

K výše uvedeným otázkám velkou měrou pomáhá užívání frameworků, a to především v oblasti responzivity a optimalizace.

1.1.2 JavaScript a TypeScript

JavaScript je programovací jazyk známý pro svou schopnost přidávat interaktivitu a dynamiku webovým stránkám. Původně byl vytvořen pro webové prohlížeče, což znamenalo, že se kód spouštěl na straně návštěvníka webu, ale dnes se používá i na serverech a v mnoha dalších prostředích (MDN Web Docs, 2024).

Mezi klíčové vlastnosti JavaScriptu patří:

- Flexibilita a univerzálnost: Může být použit pro vývoj na straně klienta (ve webovém prohlížeči) i serveru (například pomocí Node.js).
- Dynamický a interpretovaný jazyk: JavaScriptový kód se provádí za běhu, což umožňuje rychlý vývoj a snadné testování.
- Asynchronní programování: Podpora pro asynchronní funkce, jako jsou Promises a async/await, což umožňuje efektivní zpracování I/O operací.
- Event-driven (událostmi řízený) model: Velmi vhodný pro vytváření interaktivních webových aplikací.

- Podpora komunity a ekosystém: Široká podpora vývojářů, množství knihoven a frameworků (např. React, Angular, Vue.js).

TypeScript je vyvinutý společností Microsoft. Jedná se o nadstavbu JavaScriptu, což znamená, že rozšiřuje jeho možnosti. TypeScript umožňuje statické typování a další funkce, které nejsou v JavaScriptu dostupné (Typescript, 2024).

Mezi hlavní vlastnosti TypeScriptu patří:

- Statické typování: Možnost definovat typy proměnných, parametrů, návratových hodnot funkcí atd., což zvyšuje čitelnost kódu a zjednodušuje detekci chyb.
- Objektově orientované programování: Podpora pokročilejších OOP konceptů, jako jsou rozhraní a generické typy.
- Kompilace do JavaScriptu: TypeScript se kompiluje do čistého JavaScriptu, což znamená, že je kompatibilní s jakýmkoli prostředím, které podporuje JavaScript.

Hlavní rozdíly:

- Největším rozdílem mezi JavaScriptem a TypeScriptem je statické typování v TypeScriptu. Toto typování pomáhá odhalovat chyby již v průběhu vývoje a zvyšuje stabilitu a udržitelnost kódu v dlouhodobém horizontu.
- Nástrojová podpora: TypeScript obvykle poskytuje lepší nástrojovou podporu v integrovaných vývojových prostředích (IDE), což zvyšuje produktivitu vývojáře.
- Kompatibilita a flexibilita: JavaScript je více flexibilní a široce podporovaný, protože je to základní jazyk webových prohlížečů. TypeScript vyžaduje kompilaci do JavaScriptu, ale jeho další vrstva poskytuje výhody v podobě lepšího řízení kódu a optimalizace.

Nedá se jasně určit, zda je lepší JavaScript či TypeScript, oba jazyky mají svá pro a proti. V bakalářské práci využiji možnosti TypeScriptu.

1.1.3 Historie a vývoj frontendových technologií

S frontendem se lidé setkávali od samotného počátku internetu, kdy na počátku 90. let vznikl značkovací jazyk HTML, díky jemuž vznikl základ pro webové stránky. V roce 1996 bylo představeno CSS, bez kterého byly vzhledy webů jen obtížně spravovány. S nástupem JavaScriptu v druhé polovině 90. let přišla možnost tvořit weby interaktivní a dynamické. Pomohl tomu také vznik DHTML, což byla kombinace Javascriptu, HTML a CSS. Poté přišly další nadstavby JavaScriptu, jako je například Ajax a jQerry, které bylo první frontendovou knihovnou.

Kolem roku 2010 vznikl známý framework Angular, následně knihovna React a Vue.js. V roce 2021 pak přišel mocný nástroj pro tvorbu webů – Astro.

V současné době se trendy ubírají primárně směrem optimalizace pro mobilní telefony, webových aplikací a také propojení frontendu s umělou inteligencí, která otevírá spoustu nových uživatelských možností.

1.2 JavaScriptové frameworky a knihovna React

Framework je soubor knihoven, nástrojů a pokynů, který poskytuje základní strukturu pro vývoj aplikací a webů. Frameworky jsou navrženy tak, aby usnadňovaly a zrychlovaly proces vývoje

tím, že poskytují často používané funkce a řešení pro běžné problémy, čímž vývojářům zefektivňuje práci a pomáhá vyvíjet optimalizované weby.

Klíčové vlastnosti frameworků:

- Předdefinovaná struktura: Frameworky poskytují konzistentní strukturu a vzor pro aplikace. Tím pomáhají udržovat kód organizovaný a snadno spravovatelný.
- Knihovny kódu: Obsahují knihovny se znovupoužitelným kódem, což umožňuje vývojářům rychle implementovat běžné funkce bez nutnosti psát kód od základu.
- Integrace s dalšími technologiemi: Frameworky jsou často navrženy tak, aby dobře spolupracovaly s dalšími technologiemi a nástroji, což usnadňuje integraci různých aspektů vývoje aplikací.
- Zabezpečení a výkon: Frameworky mohou obsahovat vestavěná zabezpečení a optimalizace výkonu, což pomáhá při vytváření robustních a efektivních aplikací.
- Komunita a podpora: frameworky často mají aktivní komunitu vývojářů, díky čemuž je dostupná podpora, vzájemná výpomoc a notná dávka inspirace pro ideální řešení implementace.
- Komponenty a znuvupoužitelnost: Velkou výhodou je možnost vytváření si komponent, které mohou být nezávislé a znovupoužitelné na různých projektech, což zvyšuje efektivitu při práci.

1.2.1 Představení frameworků

Frontendových frameworků již existuje velké množství. Zde se však zaměříme na dva, které patří mezi nejpopulárnější a jeden, který prozatím není tak používaný, protože se jedná o relativně mladou technologii.

Důvody, proč jsem si vybral právě tyto tři framworky je ten, že Vue.js je poměrně snadný pro začátečníky, React je jedním z nejpopulárnějších a Astro je novinka, která přichází s novými možnostmi (kombinace s jinými frameworky, nulovým JavaScriptem na straně klienta) a má potenciál být nejužívanějším nástrojem pro vývoj v oblasti webů a webových aplikací.

Vue.js

Vue.js je progresivní JavaScriptový framework pro vytváření uživatelských rozhraní a byl vytvořen v roce 2014 bývalým inženýrem Google. Je známý pro svou jednoduchost, flexibilitu a intuitivní reaktivní datový model. Vue.js umožňuje snadno integrovat malé části do stávajících stránek a je také dostatečně robustní pro vývoj rozsáhlých jednostránkových aplikací (SPA).

Vue.js je vhodný pro širokou škálu webových projektů, od malých osobních stránek po rozsáhlé podnikové aplikace. Jeho popularita rychle roste, což je dáno zejména jeho přístupností pro začátečníky, flexibilitou a výkonnou komunitou, která podporuje jeho další rozvoj.

Vue.js je vyzdvihován za jeho snadné učení, přívětivost vůči začátečníkům a výkonný, ale přitom jednoduchý ekosystém. Díky tomu je oblíbenou volbou pro vývojáře, kteří hledají efektivní způsob, jak vytvářet dynamické webové aplikace (Vue.org, 2024).

React

React, známý také jako React.js nebo ReactJS, vyvinutý inženýry Facebooku v roce 2013, je knihovna pro vytváření uživatelských rozhraní, zvláště známá pro využití komponent a stavovou správu. React je velmi populární pro svou efektivitu a škálovatelnost, a je často používán pro vytváření složitých a interaktivních webových aplikací. Jeho ekosystém zahrnuje rozsáhlé množství nástrojů a knihoven.

React je široce používán v průmyslu pro vývoj dynamických a responzivních webových aplikací. Jeho flexibilita z něj činí vhodnou volbu pro malé i velké projekty. Facebook, Instagram, Airbnb a mnoho dalších velkých společností používá React pro své webové aplikace.

Reactová komunita je jednou z největších a nejaktivnějších v oblasti webového vývoje, což znamená bohatý ekosystém nástrojů, rozšíření a knihoven, jako jsou Redux pro správu stavu a Next.js pro server-side rendering a statické generování stránek.

React je ceněn pro svou rychlost, škálovatelnost a modulárnost, stejně jako pro silnou podporu ze strany komunity a průmyslu, což ho činí jedním z nejoblíbenějších nástrojů pro frontendový vývoj dneška (React.dev, 2024).

Astro

Astro je modernější framework, který se zaměřuje na vylepšený výkon webových stránek pomocí tzv. „island architecture“. Astro umožňuje vývojářům psát komponenty pomocí oblíbených frameworků jako React, Vue, nebo Svelte, a pak je renderovat na serveru pro optimalizaci výkonu. Je to přístup „zero-JavaScript by default“, kde JavaScript je posílán klientovi pouze v případě potřeby, což vede k rychlejším načítacím časům. Byl oficiálně představen v roce 2021 a od té doby získal pozornost ve světě webového vývoje díky svému unikátnímu přístupu k vytváření webových stránek a aplikací

Je vhodný pro vývojáře, kteří chtějí vytvářet rychlé, efektivní a moderní webové stránky, a pro ty, kteří chtějí experimentovat s různými frontendovými frameworky v jednom projektu.

Jeho schopnost snižovat množství posílaného JavaScriptu a využívat moderní techniky generování statických stránek činí z Astro atraktivní volbu pro projekty, kde je prioritou výkon a rychlost načítání. Astro se tedy stává zajímavou alternativou k tradičním frameworkům, zejména pro vývojáře a týmy, kteří hledají způsoby, jak zlepšit uživatelskou zkušenost prostřednictvím optimalizovaných webových aplikací (Astro.build, 2024).

1.2.2 Klíčové vlastnosti a rozdíly

Vue.js

- Přístupnost a Jednoduchost: Vue je známý svou přívětivostí vůči začátečníkům a jednoduchým, srozumitelným API. Jeho šablonová syntaxe je snadno pochopitelná, což usnadňuje učení a adaptaci.
- Reaktivní Datový Model: Vue poskytuje reaktivní a deklarativní programování, které zjednodušuje správu stavu a dat ve webové aplikaci.

- Flexibilní Ekosystém: Vue může být používán jak pro jednoduché, tak pro komplexní aplikace díky své schopnosti snadno integrovat s dalšími knihovnami a existujícími projekty.

React

- Komponentový Design: React se zaměřuje na vytváření uživatelských rozhraní pomocí komponent, což usnadňuje znovupoužitelnost kódu a správu velkých aplikací.
- Rozsáhlý Ekosystém a Komunita: React má velkou a aktivní komunitu, bohatý ekosystém nástrojů a rozšíření, včetně knihoven jako Redux pro správu stavu.
- Flexibilita a Všestrannost: React lze použít v široké škále aplikací a je dostatečně flexibilní, aby se přizpůsobil různým vývojovým potřebám.

Astro

- Flexibilita a Všestrannost: React lze použít v široké škále aplikací a je dostatečně flexibilní, aby se přizpůsobil různým vývojovým potřebám.
- Zero-JavaScript by Default: Výchozí nastavení Astro neobsahuje žádný JavaScript, což znamená, že stránky jsou lehčí a rychlejší. JavaScript je přidáván pouze tam, kde je to nutné.
- Podpora Více Frameworků: Astro umožňuje vývojářům používat komponenty z různých frontendových frameworků (např. React, Vue) v jednom projektu, což poskytuje mimořádnou flexibilitu.

Shrnutí rozdílů

Vue.js: Ideální pro ty, kdo hledají snadno použitelný a flexibilní framework s jasným programovacím modelem a rychlým startem.

React: Preferován pro vývoj komplexních aplikací s velkým množstvím dynamických dat a komponent, podporován rozsáhlou komunitou a bohatým ekosystémem.

Astro: Hodí se pro projekty, kde je prioritou výkon, rychlost načítání a potřeba integrovat komponenty z různých frontendových technologií.

1.3 Architektura a Komponenty

Ve frameworcích bývá použita specifická architektura, proto je potřeba vysvětlit její význam. Nedílnou součástí při použití frameworku jsou také komponenty. Ty jsou základem a jednou z hlavních výhod při vývoji za pomoci moderních frontendových technologií.

1.3.1 Definice a význam architektury

Definice

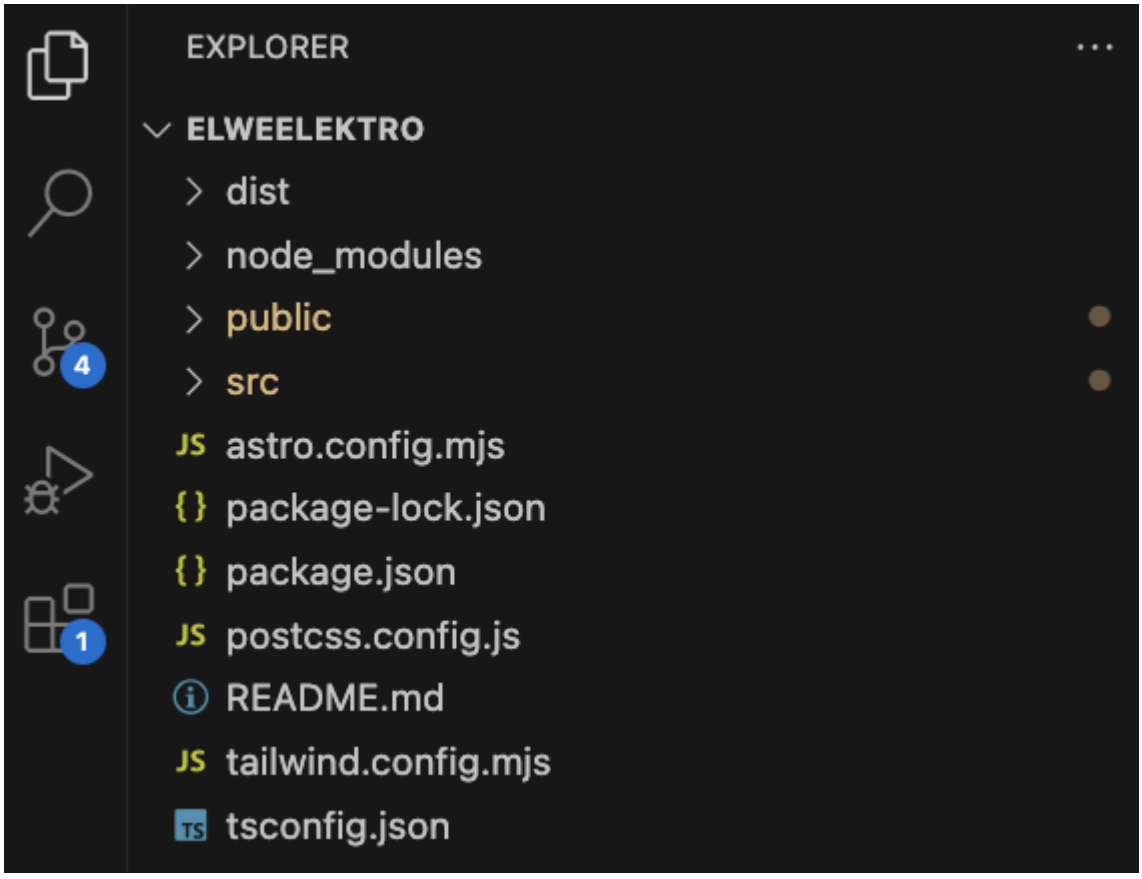
Frontendová architektura je soubor návrhových vzorů, struktur, nástrojů a procesů, které se používají při vývoji klientské strany webových aplikací. Klientská strana, často označovaná jako „frontend“, se týká všeho, co uživatel vidí a s čím interaguje ve webovém prohlížeči – od layoutu a designu až po interaktivitu a chování aplikace. Frontendová architektura se zaměřuje na

organizaci kódu, správu uživatelského rozhraní (UI), interakci s backendovými systémy a celkovou uživatelskou zkušenost.

Význam

- Organizace Kódu a Udržitelnost: Jedním z klíčových prvků frontendové architektury je udržet kód organizovaný a snadno spravovatelný. To je zvláště důležité v případě rozsáhlých projektů, kde špatně strukturovaný kód může vést k problémům s údržbou, rozšiřitelností a testováním. Důležitá je i možnost recyklace kódu v jiných projektech.
- Uživatelské Rozhraní: Správná frontendová architektura umožňuje efektivní vývoj uživatelských rozhraní, která jsou esteticky přitažlivá, intuitivní a responzivní. To zvyšuje uživatelské pohodlí a spokojenost, což je pro úspěch webových aplikací žádoucí.
- Výkon a Optimalizace: Efektivně navržená frontendová architektura zajišťuje, že aplikace bude rychlá a efektivní, což je zásadní pro poskytování hladkých a příjemných uživatelských zkušeností. To zahrnuje optimalizaci načítání zdrojů, minimalizaci latence a zajištění dobrého výkonu i na méně výkonných zařízeních.
- Integrace s backendem: V neposlední řadě musí frontend integrovat s různými backendovými službami a API. Frontendová architektura tedy musí zahrnovat strategie pro efektivní a bezpečnou komunikaci mezi klientem a serverem.

Výběr správné architektury je stěžejní pro vývoj kvalitních a uživatelsky přívětivých webů a aplikací. Při výběru správné technologie a její architektury je potřeba zvážit náročnost projektu a jeho případný růst a rozvoj.



Obrázek 1: Ukázka struktury Astro

Zdroj: vlastní tvorba

1.3.2 Definice a význam komponenty

Definice

Ve frontendovém vývoji se komponentou myslí samostatná a znovupoužitelná jednotka, která se skládá z jednoho a více prvků a tvoří část rozhraní pro uživatele. Může být chápána jako blok, tlačítko, hlavička či patička webu. Komponenta obsahuje HTML (či XML), CSS a JavaScript, popřípadě TypeScript.

Význam

- Složení uživatelského rozhraní: Komponenta je základním prvkem pro tvorbu rozhraní webových stránek a aplikací. Díky tomu lze rozhraní strukturovat do logicky oddělených částí, což zjednodušuje vývoj, údržbu, ale i přehlednost webu.
- Interaktivnost: Komponenty mohou mimo prostého zobrazení mít i interaktivní funkci, jako například tlačítko pro odeslání formuláře.
- Správa dat: U složitějších aplikací mohou být komponenty využity k uchování a zobrazení dat.
- Testování a údržba: Testování komponent může probíhat pro každou samostatně. V případě chyby pak stačí upravit komponentu a oprava se projeví všude, kde je komponenta užita.

Vlastnosti komponenty

- Samostatnost: Komponenta je vždy navržena tak, aby byla nezávislá, takže bude fungovat i bez ostatních částí webu
- Znuvupoužitelnost: Jedna z primárních výhod komponent je jejich znuvupoužitelnost v různých částech webu či dokonce napříč více projektech.
- Vlastní kód a styl: Díky vlastnímu kódu a stylu logika ani vzhled komponenty nemá vliv na zbytek webu a naopak.

1.3.3 Architektura specifických frameworků

Přístup ke komponentám se významně liší mezi Vue.js, Reactem a Astro, což ovlivňuje způsob, jakým jsou vývojáři schopni strukturovat a vyvíjet uživatelské rozhraní. Každý z těchto frameworků a knihoven nabízí unikátní nástroje pro práci s komponentami. Níže je uvedeno pár rozdílů.

Vue.js

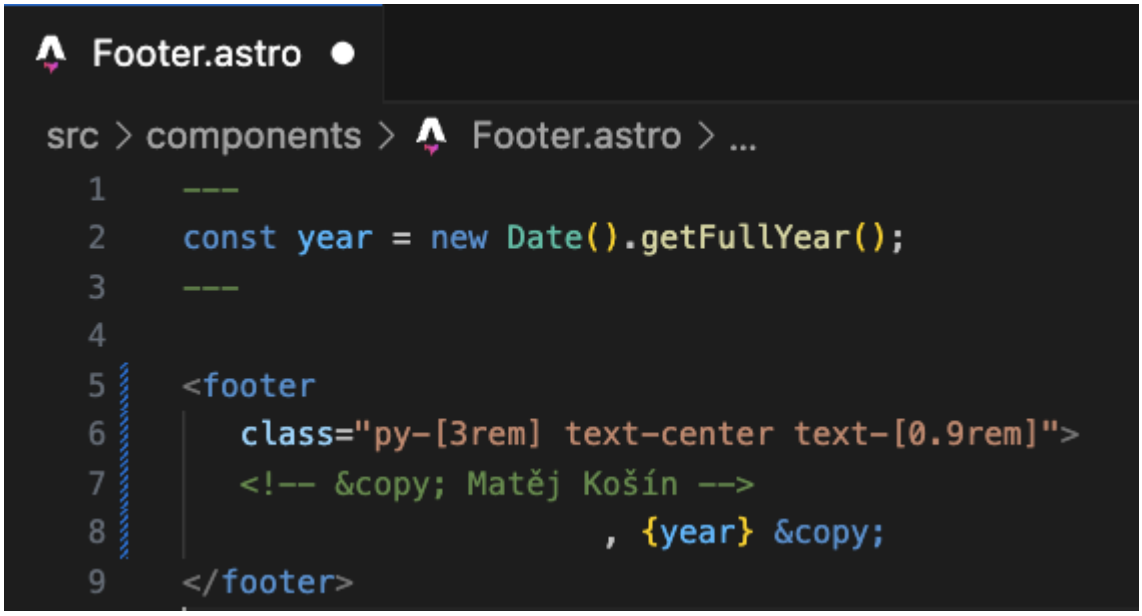
- Deklarativní Šablony: Vue.js používá HTML založené šablony, které činí strukturu komponent více deklarativní. Tyto šablony jsou snadno čitelné.
- Reaktivní Systém Dat: Vue.js poskytuje reaktivní systém pro správu dat komponent. Data ve Vue jsou reaktivní, což znamená, že Vue automaticky zjistí, když se data změní a aktualizuje DOM.

React

- JSX: React používá JSX, rozšíření syntaxe JavaScriptu, které umožňuje zapisovat HTML kód přímo v JavaScriptu. JSX zvyšuje expresivitu a moc komponent tím, že umožňuje smíšení logiky a značkování.
- Stav a Props: React používá stav (state) a props pro správu dat a interakci mezi komponentami. Stav je interní pro komponentu, zatímco props jsou způsob, jak předávat data mezi komponentami.

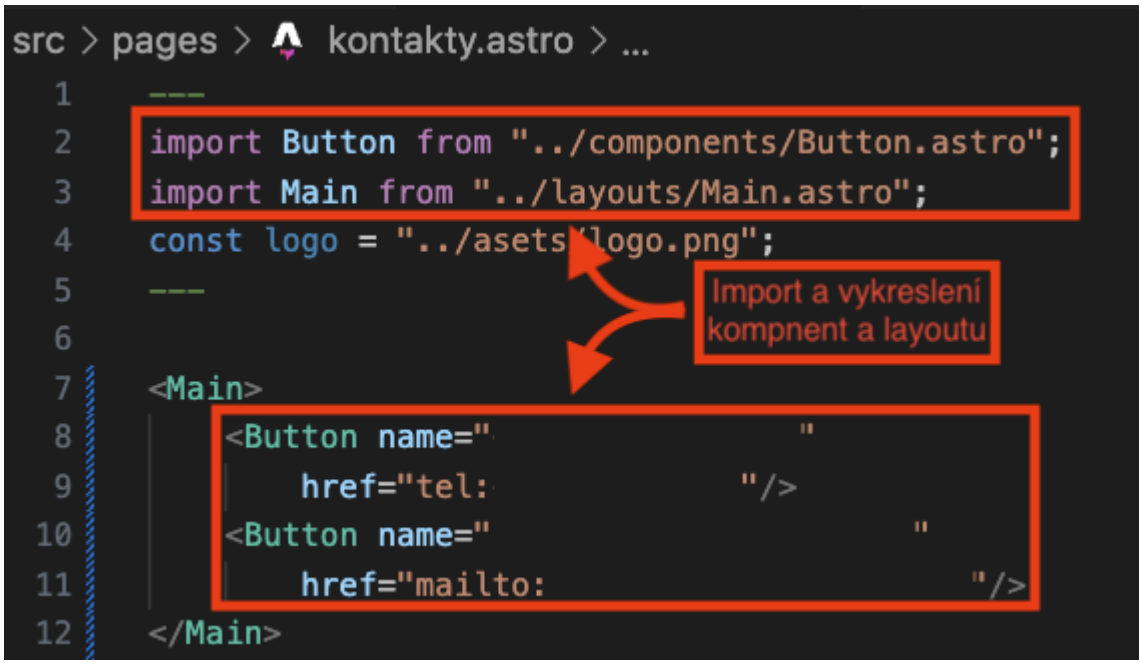
Astro

- Island Architecture: Astro přistupuje ke komponentám přes tzv. „Island Architecture“. Je to designový vzor, kde každý interaktivní prvek na stránce je izolovaným „ostrovem“, což zvyšuje efektivitu a výkon.
- Zero-JavaScript by Default: Astro v základním nastavení neodesílá JavaScript klientovi, pokud to není explicitně vyžadováno. To je v kontrastu s Reactem a Vue, kde JavaScript hraje klíčovou roli. Díky tomu se rychlost načtení webu neomezuje výkonem zařízení a rychlostí internetu návštěvníka.
- Podpora Více Frameworků: Astro umožňuje vývojářům používat komponenty z různých frontendových frameworků (jako jsou React, Vue.js, Solid.js) v rámci jednoho projektu.



Obrázek 2: Ukázka komponenty Astro

Zdroj: vlastní tvorba



Obrázek 3: Použití komponent Astro

Zdroj: vlastní tvorba

1.4 React nebo Astro

V bakalářské práci se primárně zaměřuji na React a Astro, a to z důvodu, že React je považován na nejoblíbenější framework v roce 2023 a Astro je jeden z nejmladších frameworků a má svá specifika, která mě velmi zaujala.

React je vhodným nástrojem pro vytvoření interaktivní a dynamické aplikace, zatím co Astro je vyvinuto spíše na statické stránky, jako jsou třeba blogy a portfolia, aby co nejvíce optimalizovalo výkon a rychlost.

1.5 CSS framework Tailwind

Místo klasického CSS je vhodné využívat CSS frameworky, jako je například Tailwind. Důvodů je hned několik, především však rychlejší vývoj, přizpůsobitelnost a snaží tvorba responzivity.

Vlastnosti

- Utility-First Přístup: Na rozdíl od tradičních CSS frameworků, jako je Bootstrap, které poskytují předem definované komponenty, Tailwind CSS se zaměřuje na poskytování utility tříd. Tyto třídy umožňují vývojářům rychle a efektivně stylizovat elementy bez nutnosti psát vlastní CSS. Například místo definování vlastních tříd pro tlačítka, můžete v Tailwindu použít kombinaci utility tříd pro nastavení barvy, paddingu, marginu atd.
- Přizpůsobitelnost: Tailwind CSS je vysoce přizpůsobitelný. Vývojáři mohou definovat vlastní barvy, velikosti a další prvky ve svém konfiguračním souboru, což umožňuje vytvářet jedinečné designy a zachovávat konzistenci stylů napříč projektem.
- Responzivní Design: Tailwind CSS usnadňuje implementaci responzivního designu pomocí utility tříd pro různé velikosti obrazovky a zařízení.

- Just-In-Time (JIT) Kompilace: JIT režim v Tailwind CSS generuje styly na vyžádání, což znamená, že ve vývoji jsou vytvářeny pouze ty CSS třídy, které jsou skutečně použity. To vede k menším velikostem souborů a rychlejšímu vývoji.

Tailwind CSS je oblíbený mezi vývojáři pro svou flexibilitu, efektivitu a schopnost rychle vytvářet uživatelská rozhraní bez nutnosti psát rozsáhlé množství vlastního CSS. Jeho utility-first přístup a přizpůsobitelnost dělají z Tailwindu mocný nástroj pro moderní webový vývoj. Nicméně, pro nováčky může být Tailwind CSS náročnější na učení kvůli potřebě seznámit se s velkým množstvím utility tříd a způsobem jejich použití (MICHÁLEK, Martin. *Tailwind CSS, 2021*).

2 Praktická část

2.1 Úvod do praktické části

Praktická část zahrnuje vývoj jednoduché webové aplikace, která napomůže základnímu porovnání dvou vybraných frameworků, a to React.js a Astro. Jako vzorovou aplikaci jsem vybral kalkulačku se základními funkcemi, protože se jedná o výtečný nástroj pro každodenní užití.

Hlavním cílem je porovnání, jak se s frameworky pracuje z pohledu vývojáře, optimalizace kódu, náročnosti na výkon a rychlost načítání.

Aplikace je programována ve freeware editoru zdrojového kódu Visual Studio Code na platformě macOS. Projekt bude verzován pomocí desktopové aplikace GitHub a synchronizován na platformě GitHub.

2.2 Aplikace

2.2.1 Návrh aplikace

Jak již bylo zmíněno v úvodu do praktické části, implementovanou aplikací je kalkulačka se základními výpočetními operacemi sčítáním, odčítáním, násobením a dělením. V rámci programové části kalkulačky jsou i ošetřeny výpočetní omezení, jako je například nemožnost dělení nulou.

UI kalkulačky

Uživatelské prostředí aplikace je navrženo s ohledem na co nejvyšší přehlednost a ovladatelnost. Aplikace má pouze jednu plochu, kde je pole velkých tlačítek s čísly, matematickými operacemi a textový řádek, kde se zobrazuje zadaný příklad k vypočítání. Následně velké tlačítko s operátorem přiřazení „=“, po jehož stisknutí se v dalším textovém poli zobrazuje výsledek. Ve výčtu tlačítek je možnost obnovení obsahu kalkulačky, které umožňuje zadání nového příkladu. V případě, že se uživatel pokouší dělit nulou, aplikace ho upozorňuje hláškou, že nelze dělit nulou.

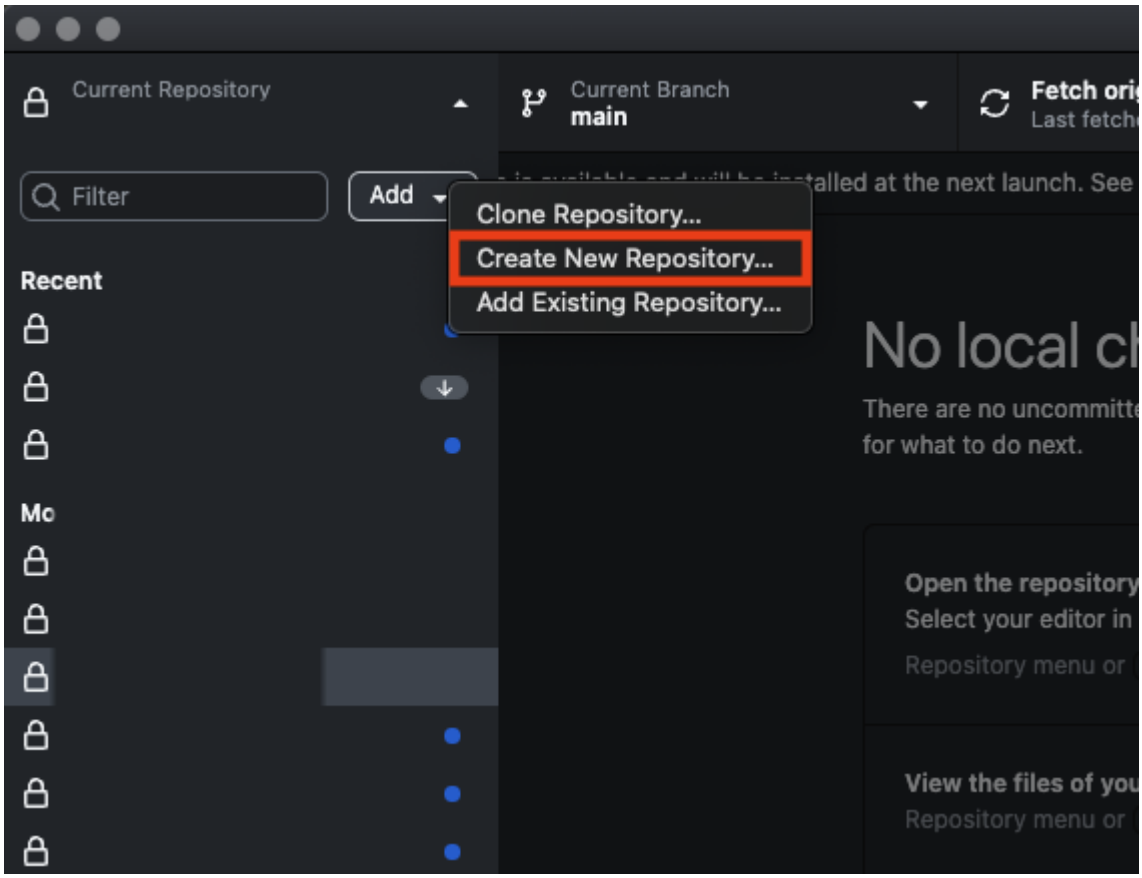
Funkce kalkulačky

1. Sčítání
   1.1. Tlačítko se symbolem „+“.
   1.2. Provede sečtení zadaných čísel.
2. Odčítání
   2.1. Tlačítko se symbolem „-“.
   2.2. Provede odečtení zadaných čísel.
   2.3. Lze jít do záporných čísel.
3. Násobení
   3.1. Tlačítko se symbolem „*“.
   3.2. Provede součin zadaných čísel.
4. Dělení
   4.1. Tlačítko se symbolem „/“.
   4.2. Vydělí zadaná čísla.
   4.3. Nelze dělit nulou.
5. Obnovení
   5.1. Tlačítko se symbolem „C“ jako clear, v překladu vyčistit.
   5.2. Smaže obsah kalkulačky zadaný uživatelem a připraví kalkulačku k dalšímu užití.

2.3 Založení repositáře

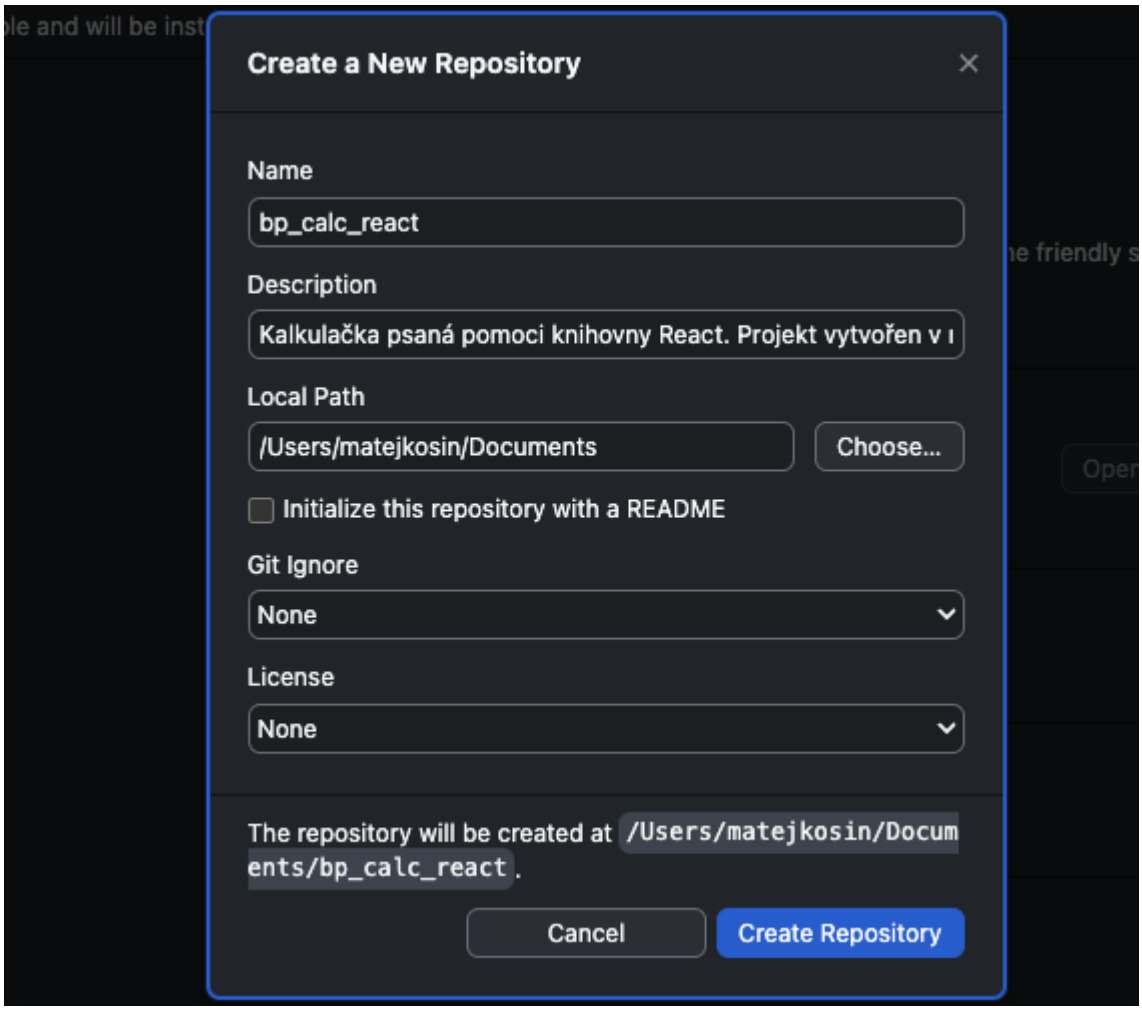
Moderním postupem ve vývojářských týmech je verzování projektů pomocí verzovacích softwarů. Verzování umožňuje uchovávat historii změn napříč celým projektem, to usnadňuje případný návrat k dřívějším verzím, zefektivňuje práci v týmu a zálohuje kód. Můžeme navíc hodnotit postup vývoje, navrhovat zlepšení v případných budoucích projektech a v neposlední řadě pomocí verzovácích softwarů lze automatizovat nasazování aplikací do produkčního prostředí.

Pro usnadnění práce používám aplikaci GitHub Desktop, kterou mám propojenou se svým účtem na platformě GitHub.



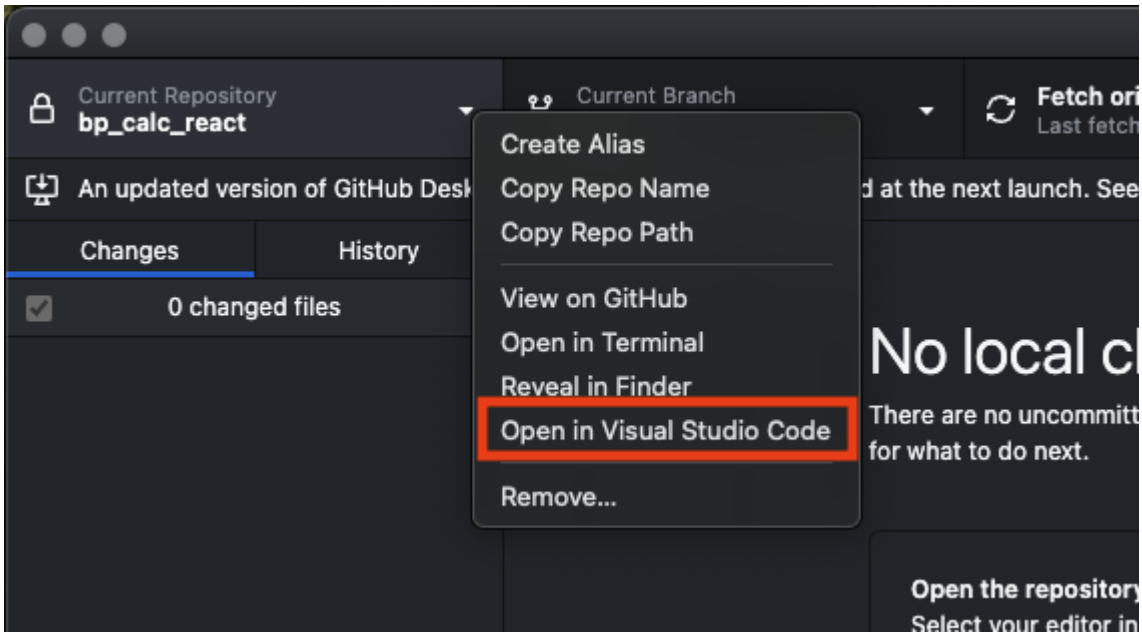
Obrázek 4: Vytvoření repositáře v aplikaci GitHub desktop

Zdroj: vlastní tvorba



Obrázek 5: Vyplnění údajů projektu
Zdroj: vlastní tvorba

Po založení repositáře si projekt lze otevřít v editoru kódu Visual Studio Code.



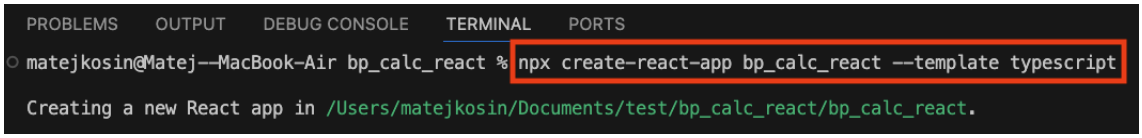
Obrázek 6: Otevření projektu v editoru

Zdroj: vlastní tvorba

2.4 Implementace v React.js

2.4.1 Vytvoření projektu

Vytvoření prázdného projektu provedeme v terminálu editoru pomoci příkazu npx create-react-app nazev_projektu --template typescript. Část „--template typescript“ nám do projektu automaticky nainstaluje typescrtipt místo javascriptu.

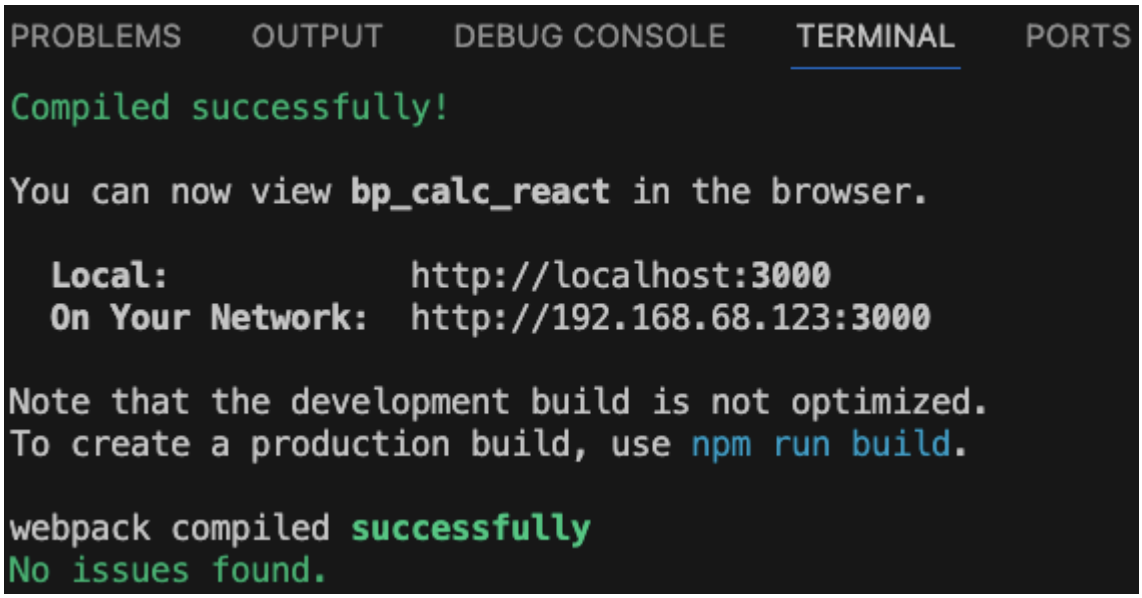


Obrázek 7: Vytvoření projektu pomocí knihovny React

Zdroj: vlastní tvorba

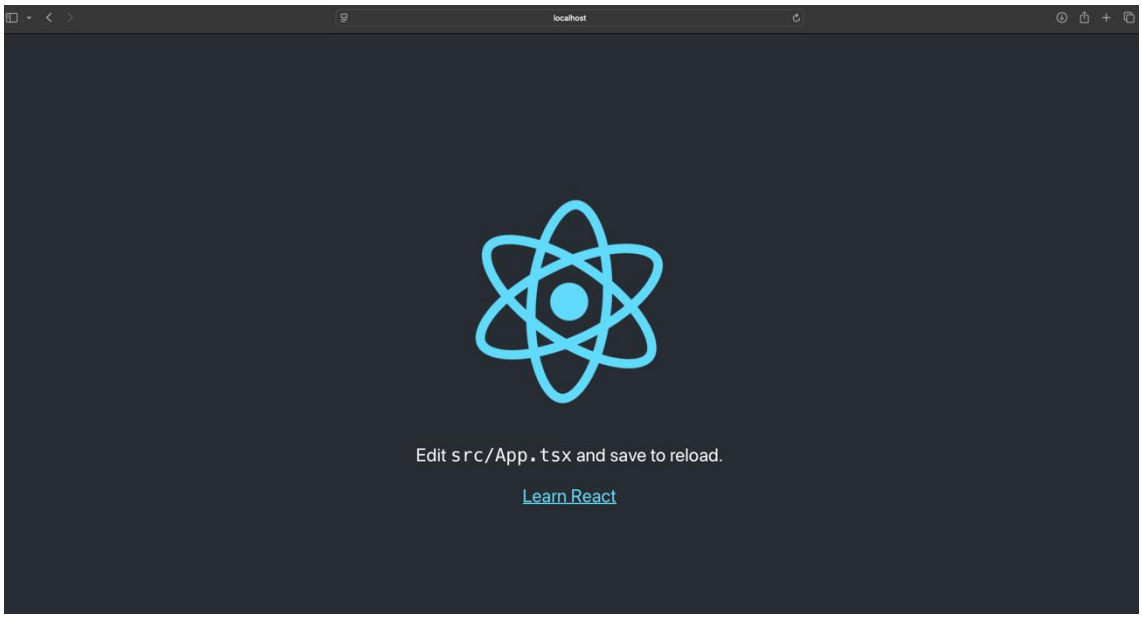
Po úspěšné instalaci se ve struktuře projektu objeví nový adresář s názvem naší aplikace. Následné spuštění aplikace v lokálním prostředí provedeme sérií příkazů v terminálu následovně:

- cd bp-calc-react – Příkaz nás přesune do složky projektu.
- npm start – Lokálně spustí projekt a otevře ve výchozím prohlížeči na adrese http://localhost:3000.



Obrázek 8: Úspěšné sestavení projektu na lokálním prostředí

Zdroj: vlastní tvorba



Obrázek 9: Prázný projekt v knihovně React

Zdroj: vlastní tvorba

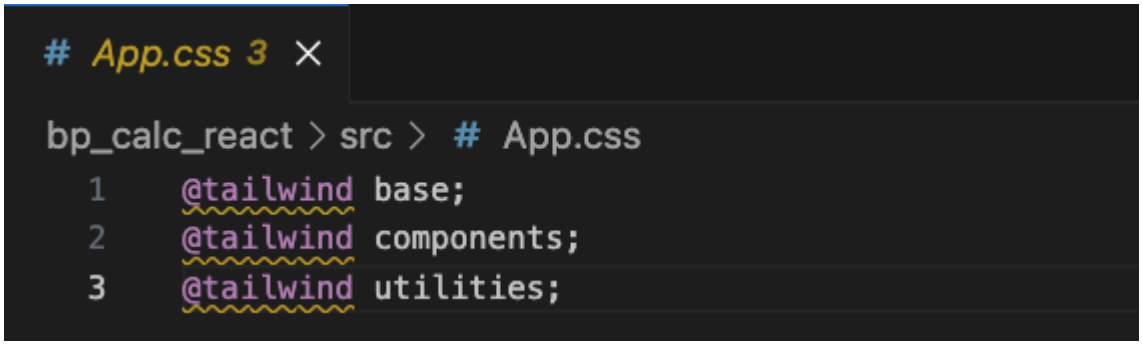
Lokálně spuštěná aplikace může běžet i během jejího vývoje a lze v reálném čase sledovat úpravy stylů, kdy stačí provedenou úpravu uložit a aplikace se v prohlížeči automaticky překreslí. Tato vlastnost vývoje React aplikace zefektivňuje vývoj a stylování aplikace.

2.4.2 Instalace Tailwind CSS

Pro úpravu vzhledu aplikace použiji CSS framework Tailwind, který je potřeba do projektu doinstalovat terminálovými příkazy:

- npm install -D tailwindcss postcss autoprefixer – Nainstaluje Tailwind CSS a potřebné balíčky jako závislosti pro vývoj.
- npx tailwindcss init -p – Inicializuje vytvoření konfiguračních souborů tailwind.config.js a postcss.config.js.

Do hlavního CSS souboru projektu, v tomto případě App.css je potřeba vepsat tyto tři řádky, které umožňují přidat různé úrovně stylů do projektu. (Tailwindcss, 2023)



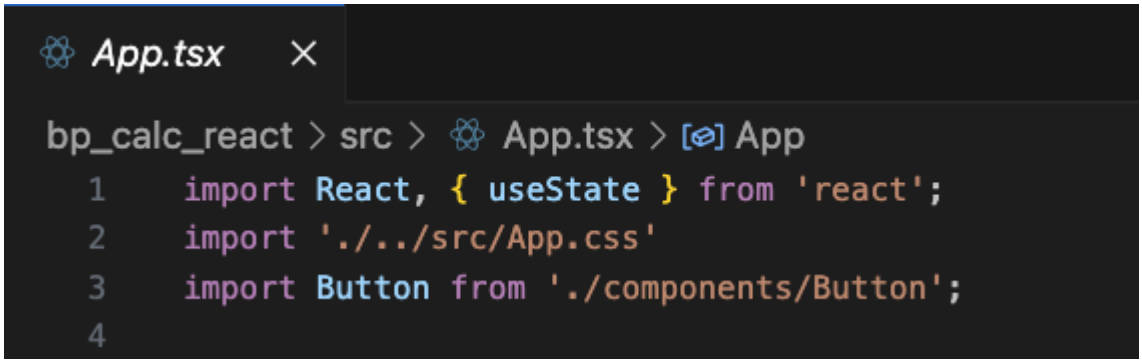
Obrázek 10: CSS soubor
Zdroj: vlastní tvorba

2.4.3 Popis kódu

V rámci kapitoly je proveden rozbor některých řádků kódu souboru App.tsx, který je srdcem celé aplikace a souboru Button.tsx, který je zástupcem komponent, pomocí kterých jsou react aplikace tak populární a udržitelné.

App.tsx

Na začátku každého tsx React souboru je nutné provést import samotného Reactu, useState, který umožňuje spravovat stav komponenty. Dále importujeme CSS soubor a komponentu tlačítka.



Obrázek 11: Import

Zdroj: vlastní tvorba

Definice funkční komponenty s použitím typescriptu je provedena na řádku 5. Řádky 6–13 určeny k deklaraci stavů komponenty.

- dispVal – Hodnota, která je aktuálně zobrazená na kalkulačce.
- prevVal – Hodnota předchozího čísla pro operace.
- operator – Slouží k uložení zvoleného operatoru.
- error – Ukládá možnou chybovou hlášku.

Následují deklarace proměnných pro hodnoty tlačítek a jejich styly.

- buttons – Pole s hodnotami pro tlačítka.
- mainButtStyle – Třídy Tailwind CSS pro tlačítka kalkulačky.
- delButtStyle – Třídy pro mazací tlačítko.

```
5   const App: React.FC = () => {
6     const [dispVal, setDispVal] =
7       useState<string>("0");
8     const [prevVal, setPrevVal] =
9       useState<string | null>(null);
10    const [operator, setOperator] =
11      useState<string | null>(null);
12    const [error, setError] =
13      useState<string | null>(null);
14
15    const buttons = ["1", "2", "3", "+", "4", "5", "6",
16                     "7", "8", "9", "*", "0", ".", "=",
17    const mainButtStyle =
18      "w-[20%] border rounded-lg shadow-sm shadow-grey
19    const delButtStyle =
20      "w-[27.5rem] border-2 rounded-lg shadow-sm shadow-
```

Obrázek 12: Deklarace stavů a proměnných

Zdroj: vlastní tvorba

Funkce „handleClick“ reaguje na kliknutí tlačítek. Vyskytne-li se chyba, při dalším kliknutím se resetuje a zobrazená hodnota bude opět „0“. Po nastavení prvního čísla a vybrání požadované

ho operátoru se první hodnota uloží do „prevVal“ a operátor do „operator“. Kliknutí na „=“ zavolá funkci „calculate“, která provede výpočet a následně zaktualizuje zobrazenou hodnotu ve stavu „dispVal“.

```
18    const handleClick = (value: string) => {
19      if (error) {
20        setError(null);
21        setDispVal("0");
22      }
23
24      if (["+", "-", "*", "/"].includes(value)) {
25        setOperator(value);
26        setPrevVal(dispVal);
27        setDispVal("0");
28      } else if (value === "=") {
29        const result = calculate();
30        setDispVal(result);
31        setPrevVal(null);
32        setOperator(null);
33      } else {
34        setDispVal(
35          dispVal === "0" ? value : dispVal + value);
36      }
```

Obrázek 13: Funkce handleClick

Zdroj: vlastní tvorba

Další funkcí aplikace je samotná kalkulace výsledku „calculate“. Ta provádí matematické operace na základě uživatelem zvoleného operátoru uloženého ve stavu „operator“. V první řadě funkce zkontroluje, zda „prevVal“ a „operator“ nejsou prázdné a jestli se uživatel nepokouší dělit nulou. Po splnění těchto podmínek vytvoří proměnnou číselného typu a dle zvolené operace provede výpočet, jehož postup má definován v přepínači jednotlivými možnými případy. Na závěr vrátí proměnnou s výsledkem.

```
43    const calculate = (): string => {
44      if (!prevVal || !operator) return dispVal;
45
46      const prev = parseFloat(prevVal);
47      const current = parseFloat(dispVal);
48
49      if (operator === "/" && current === 0) {
50        setError("Nelze dělit nulou");
51        return "0";
52      }
53
54      let result: number;
55      switch (operator) {
56        case "+":
57          result = prev + current;
58          break;
59        case "-":
60          result = prev - current;
61          break;
62        case "*":
63          result = prev * current;
64          break;
65        case "/":
66          result = prev / current;
67          break;
68        default:
69          return dispVal;
70      }
71      return result.toString();
72    };
```

Obrázek 14: Funkce calculate

Zdroj: vlastní tvorba

Poslední funkcí aplikace je „handleClear“, která po inicializaci resetuje kalkulačku. Nastaví stavy na hodnotu „null“ a „dispVal“ na nulu.

```
74    const handleClear = () => {
75      setDispVal("0");
76      setPrevVal(null);
77      setOperator(null);
78      setError(null);
79    };
```

Obrázek 15: Funkce pro reset kalkulačky

Zdroj: vlastní tvorba

Předposlední částí kódu souboru je vykreslení komponenty pomocí syntaxe TSX, která je velmi podobná syntaxi HTML. Řádek kódu 89 kontroluje, zda „error“ obsahuje nějakou hodnotu. Pokud ano, vypíše ji. V opačném případě, kdy je „error“ prázdný, zobrazí „dispVal“. Kód obsahuje funkci „map“, která iteruje pole „buttons“ a jednotlivé prvky ukládá do proměnné „btn“, která je následně použita jako hodnota komponenty tlačítka. Řádky 93 a 98 jsou komponenty Button.tsx.

```
81    return (
82      <div className='flex items-center justify-center h
83        <div className="flex items-center justify-center
84          border-2 rounded-lg shadow-lg shadow-black p-5
85          <h2 className='text-2xl font-bold uppercase m-
86            kalkulačka
87          </h2>
88          <div className="w-[27.5rem] border-2 rounded-l
89            {error ? error : dispVal}
90          </div>
91          <div className="flex flex-wrap items-center ju
92            {buttons.map((btn) => (
93              <Button key={btn} value={btn}
94                onClick={handleClick}
95                className={mainButtStyle}/>
96            ))}
97          </div>
98              <Button value="Vymazat"
99                onClick={handleClear}
100               className={delButtStyle}/>
101       </div>
102     </div>
103   );
104   };
```

Obrázek 16: Vykreslení komponenty

Zdroj: vlastní tvorba

Na závěr je proveden export komponenty, aby ji bylo možné použít jinde. V tomto případě se jedná o hlavní komponentu App.tsx, která je poté použita pouze v souboru index.tsx, kde je renderovaná jako kořenová komponenta aplikace.

```
105
106   export default App;
```

Obrázek 17: Export komponenty

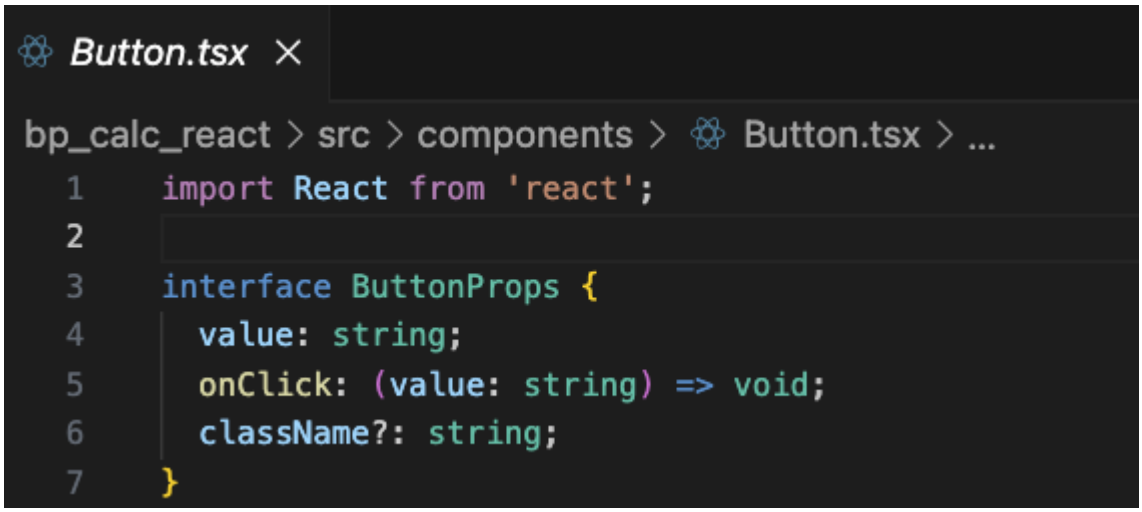
Zdroj: vlastní tvorba

Button.tsx

Import v komponentě obsahuje pouze React, protože ostatní závislosti podědí od prvku, kde bude použita, v tomto případě od App.tsx.

Od třetího řádku je definice interface ButtonProps, které popisuje vlastnosti komponenty. Vlastnosti použité v tlačítku jsou:

- value – Povinná vlastnost typu textového řetězce, která bude použita jako text v tlačítku.
- onClick – Povinná funkce tlačítka, která se spustí po jeho stisknutí a přijímá argument typu string. Do argumentu se po přivolání komponenty vloží funkce, kterou má tlačítko vykonávat (viz. Obrázek 16, řádky kódu 94 a 99).
- className – Nepovinná vlastnost, což nám určuje otazník za názvem vlastnosti, typu textového řetězce, která může obsahovat názvy tříd pro dané tlačítko.



Obrázek 18: ButtonProps
Zdroj: vlastní tvorba

Button je funkční komponenta definovaná jako React.FC<ButtonProps>, což znamená, že komponenta přijímá vlastnosti popsané rozhraním ButtonProps. V parametrech { value, onClick, className } se de-strukturalizují props pro snadnější použití.

Komponenta vrací HTML button element s atributy:

- className={className} – Nastaví Tailwind CSS třídu z „className“ vlastnosti, pokud je poskytnuta.
- onClick={() => onClick(value)} – Nastaví funkci „onClick“, která se volá při kliknutí na tlačítko. Funkce obaluje volání „onClick(value)“, což zajistí, že při kliknutí se zavolá funkce s „value“ jako argumentem.
- {value} – Obsah tlačítka. Hodnota „value“ se zobrazí jako text uvnitř tlačítka.

Na konci kódu je opět proveden export komponenty pro další použití.

```
9   const Button: React.FC<ButtonProps> =
10        ({ value, onClick, className }) => {
11          return (
12            <button
13              className={className}
14              onClick={() => onClick(value)}
15            >
16              {value}
17            </button>
18          );
19  };
20
21  export default Button;
```

Obrázek 19: Vykreslení a export komponenty tlačítka

Zdroj: vlastní tvorba

Index.tsx

Soubor index.tsx je hlavním vstupním bodem aplikace vytvořené pomocí Reactu. Slouží k nastavení a vykreslení primární komponenty „App.tsx“ do HTML stránky.

V importech je navíc, oproti předchozím komponentám, uveden „ReactDom“, pomocí kterého je možné vykreslit aplikaci do DOM, a hlavní komponenta „App“.

Document.getElementById('root') najde HTML element s ID root, který je definován v souboru index.html. Tento element slouží jako kořen aplikace, kam se celá React aplikace vykreslí. „as HTMLElement“ je typescriptová syntaxe pro ujištění, že „getElementById“ vrátí HTML Element. Pokud není prvek nalezen, vyhodí TypeScript chybu. „ReactDOM.createRoot(...)“ pak vytváří root, který umožňuje použití nových funkcí React.

Následný řádek 8 spustí renderování aplikace. Render obsahuje striktní mód, který zachycuje potenciální chyby a problémy aplikace. Uživatele sestavené aplikace nijak neovlivní, avšak ve vývojovém prostředí vypisuje varování o nepodporovaných či zastaralých funkcích. Na řádku 10 vykreslujeme samotnou hlavní komponentu.

index.tsx

bp_calc_react > src > index.tsx > ...

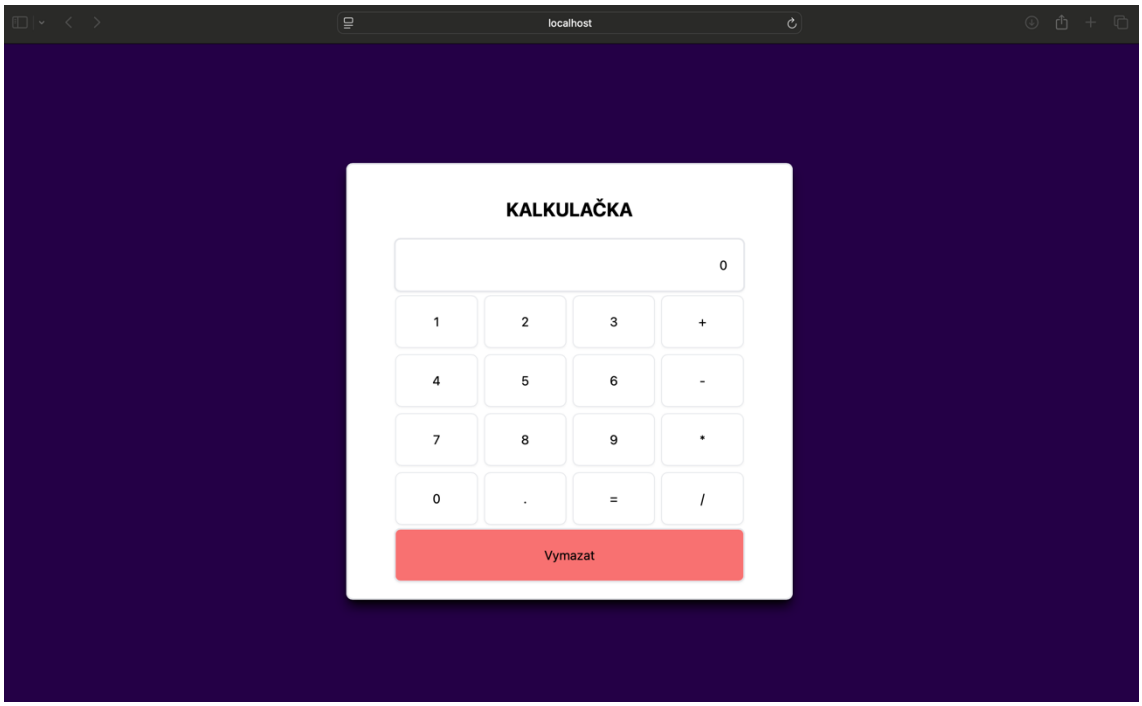
```tsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
);
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

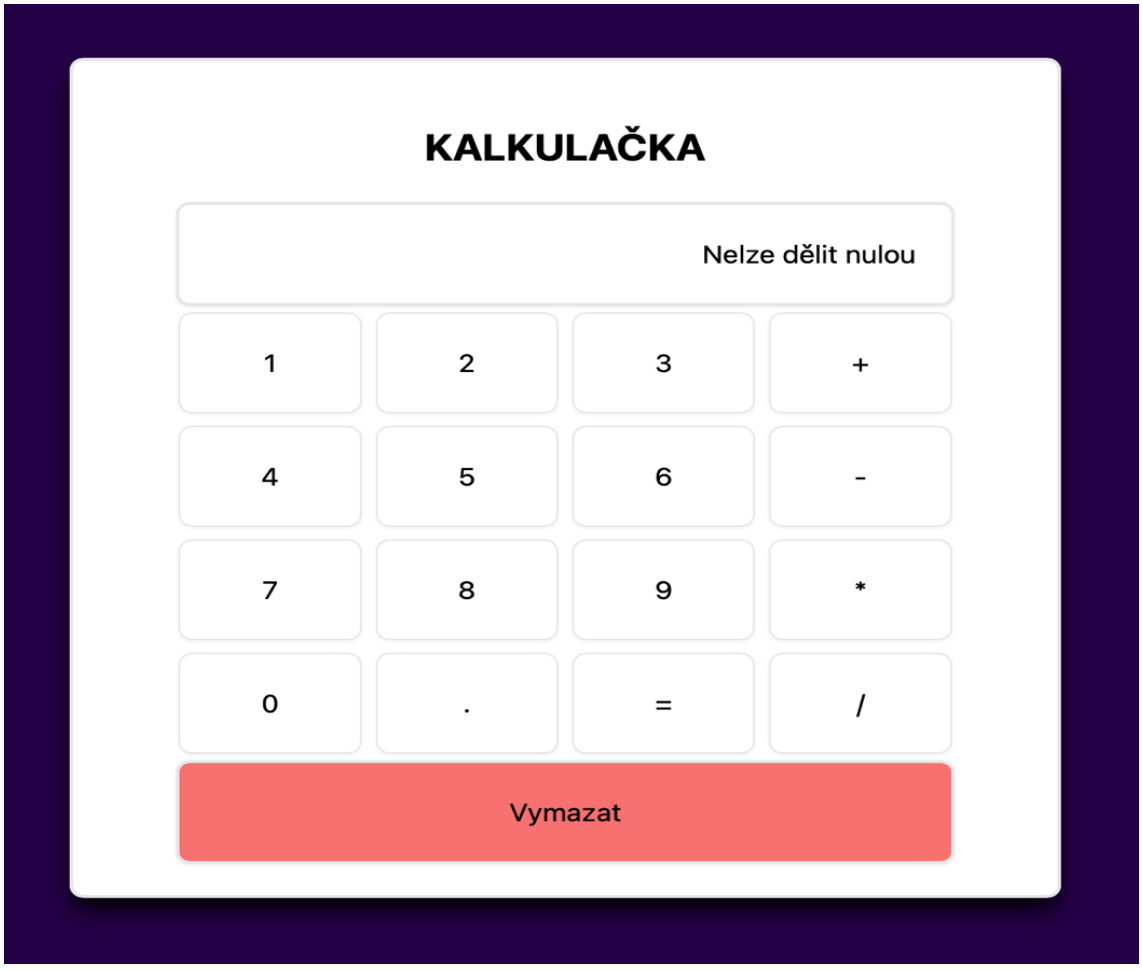
Obrázek 20: Index.tsx
Zdroj: vlastní tvorba

2.4.4 Ukázka aplikace

Aplikace je veřejně dostupná na adrese: http://bp-calc-react.netlify.app.



Obrázek 21: Ukázka aplikace
Zdroj: vlastní tvorba



Obrázek 22: Hláška, kdy se uživatel pokusil dělit nulou
Zdroj: vlastní tvorba

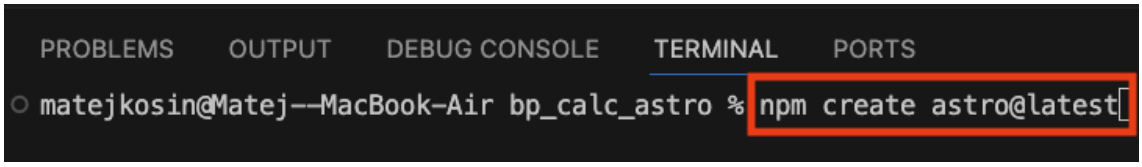
2.5 Implementace v Astro

Astro neumožňuje použití JavaScriptu na straně uživatele. Kalkulačka proto nemůže fungovat bez integrace s jiným frameworkem, který umožňuje interaktivitu na straně klienta. V druhém projektu tedy používám Astro v kombinaci s Reactem, což vytvoří silnou kombinaci dvou frameworků.

Pro projekt byl opět vytvořen repositář na platformě GitHub.

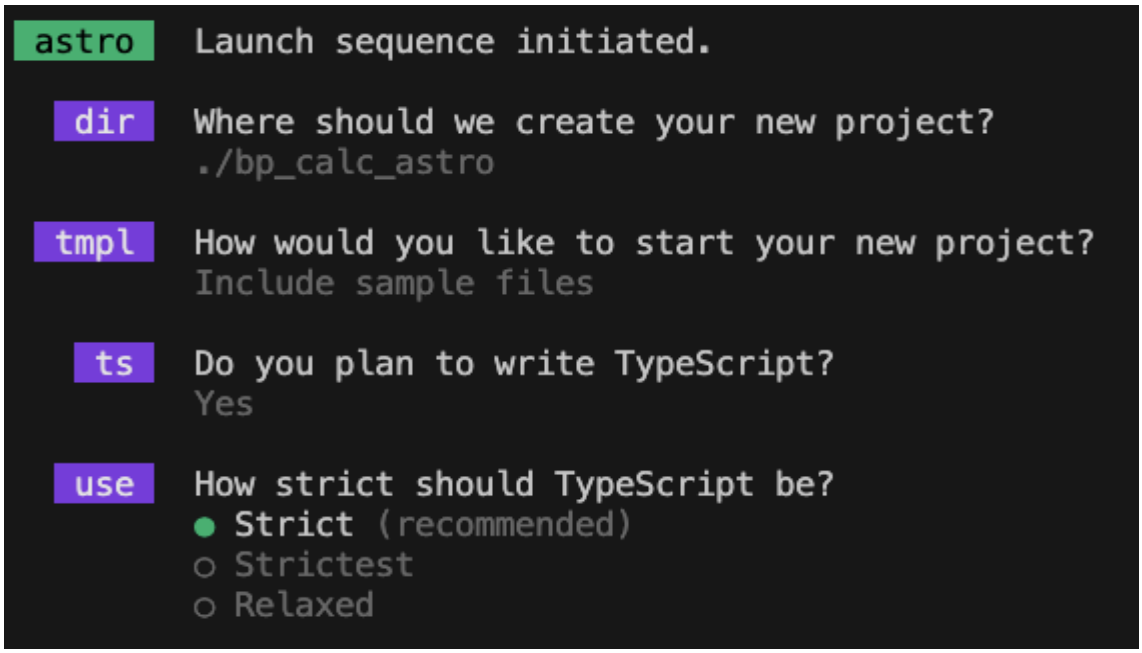
2.5.1 Vytvoření projektu

Projekt ve frameworku Astro vytvoříme pomocí terminálového příkazu „npm create astro@latest“, po kterém jsme vyzváni k základnímu nastavení projektu. Možnosti, které při zakládání aplikace můžeme zvolit jsou adresář, kde bude projekt vytvořen, zda se má implementovat nějaká šablona, či prázdný prostor a zda chci použít typescript, případně, zda striktní nebo ne. Pro aplikaci kalkulačky byl zvolen striktní typescrpit.



Obrázek 23: Příkaz k vytvoření projektu v Astro

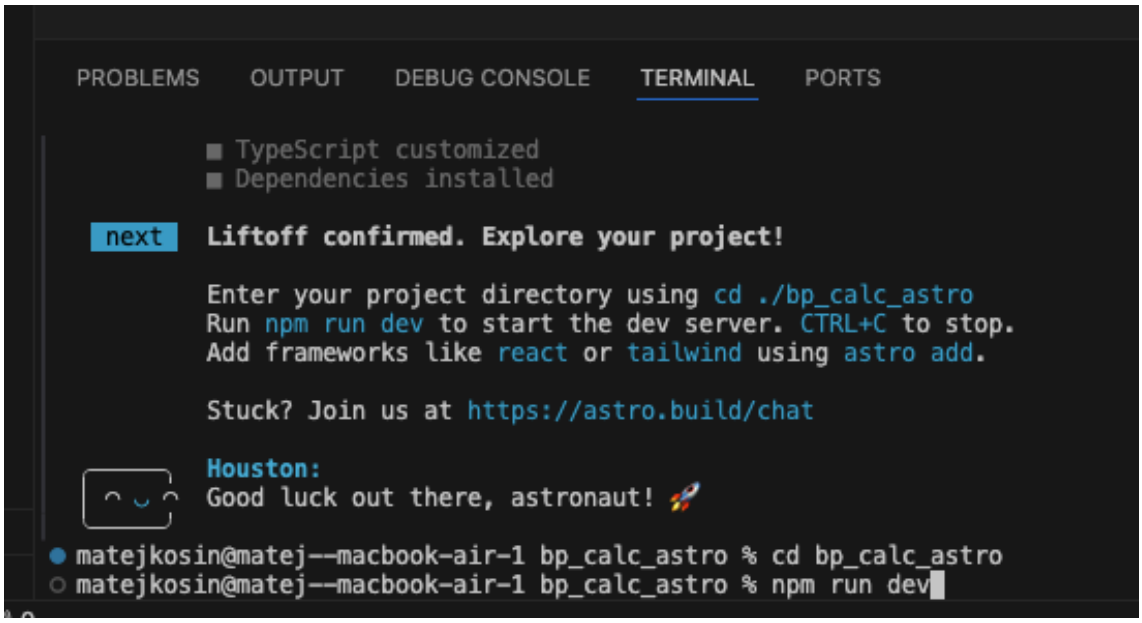
Zdroj: vlastní tvorba



Obrázek 24: Nastavení projektu

Zdroj: vlastní tvorba

Po zadání a spuštění série příkazů „cd bp_calc_astro“ a „npm run dev“ bude právě vytvořená aplikace dostupná na lokálním portu, který se v terminálu vypíše. V prohlížeči je opět možné sledovat uložené úpravy kódu v reálném čase bez nutnosti obnovení stránky.

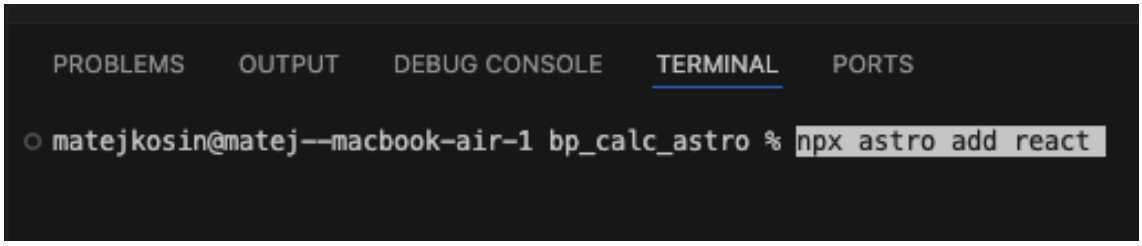


Obrázek 25: Informace, že se projekt založil a série spouštěcích příkazů

Zdroj: vlastní tvorba

2.5.2 Instalace frameworku React a Tailwind CSS

Abychom v projektu Astro mohli použít komponentu Reactu, musíme provést jeho instalaci příkazem „npx astro add react“. Příkaz doinstaluje potřebné balíčky a závislosti.



Obrázek 26: Příkaz k instalaci Reactu

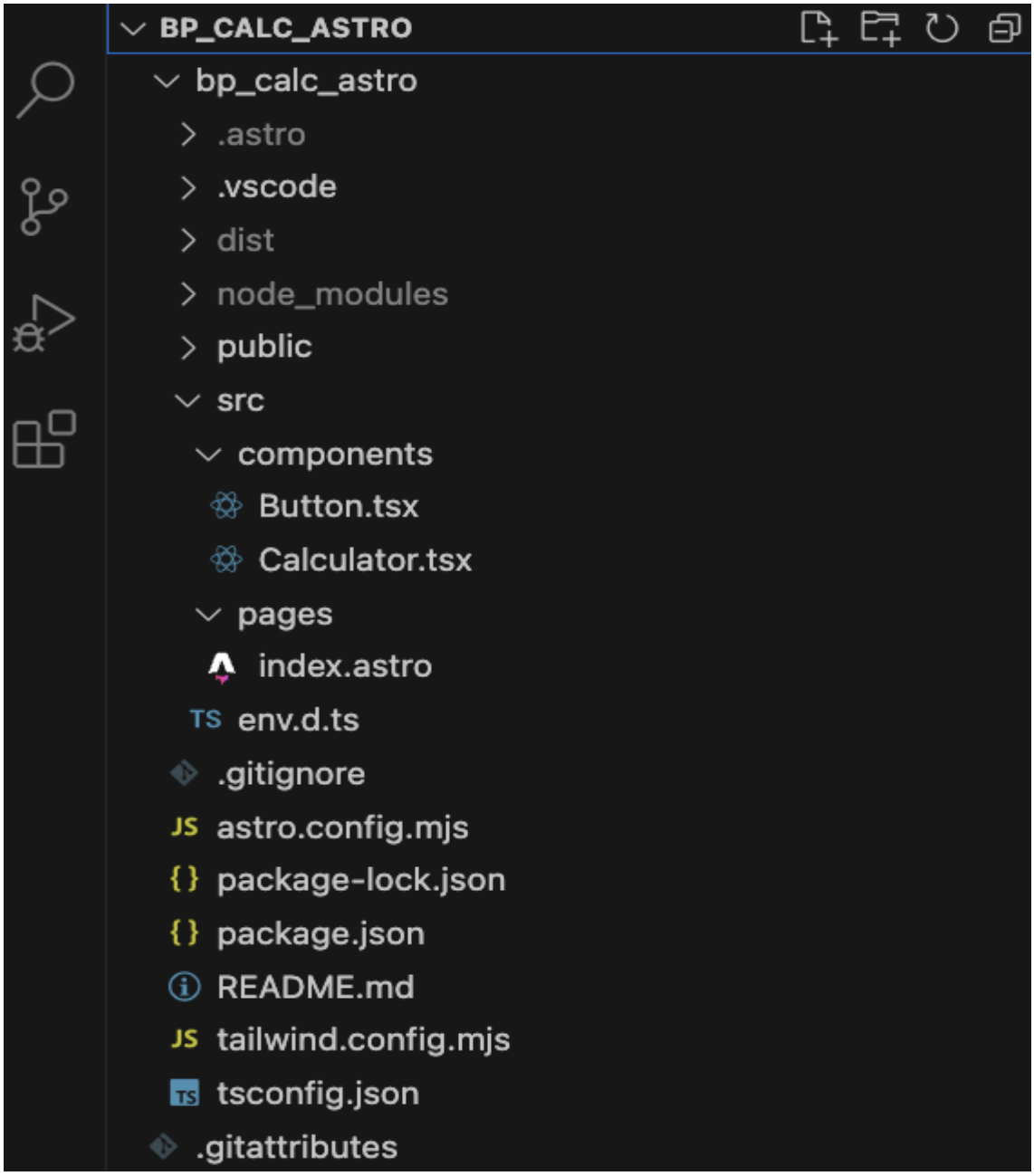
Zdroj: vlastní tvorba

Tailwind CSS, který je v aplikaci opět použit jako framework kaskádových stylů, se do projektu implementuje terminálovým příkazem „npx astro add tailwind“. Pro zprovoznění CSS frameworku není nic dalšího potřeba nastavit a lze jej začít používat.

2.5.3 Popis kódu

Jak již bylo zmíněno, astro neumožňuje použití javascriptu na straně uživatele. Použijeme tedy pro implementaci kalkulačky React, což krásně demonstruje možnosti a efektivitu používání frontendových frameworků. Místo programování nové kalkulačky lze jednoduše vzít již vytvořenou komponentu kalkulačky a tlačítka z projektu v Reactu.

Z původního projektu, který byl napsán v rámci této práce, stačí zkopírovat pouhé dva soubory a vložit je do složky „components“ v adresáři projektu Astro. Využitými soubory jsou „App.txt“ a „Button.txt“, přičemž komponentu „App“ pouze přejmenujeme na „Calculator“, aby struktura projektu byla co nejpřehlednější.



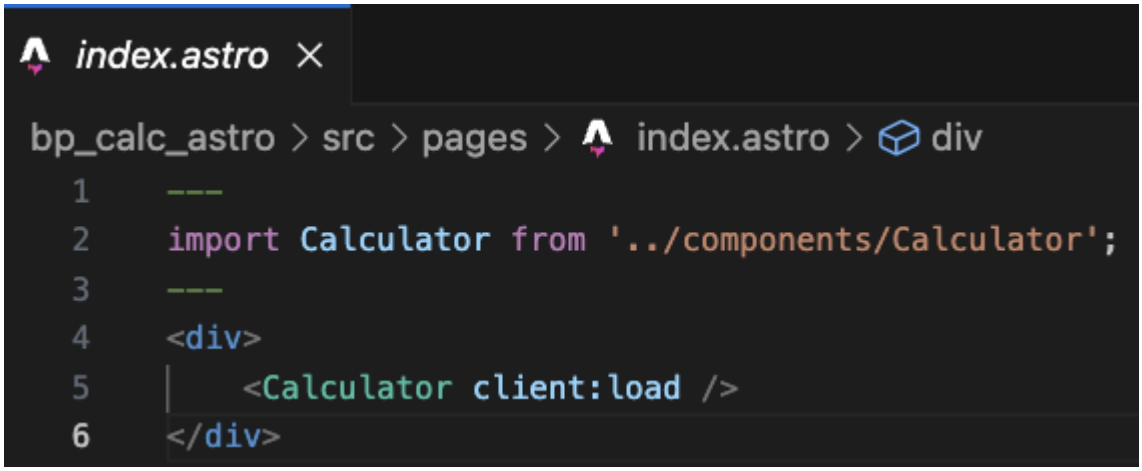
Obrázek 27: Adresářová struktura Astro s využitím Reactu

Zdroj: vlastní tvorba

Index.astro

„Index.astro“ je v přirovnání s Reactem jako soubor „App.txt“, tudíž základní kámen projektu. V jeho kódu musíme provést implementaci komponenty kalkulačky, které však musíme přiřadit vlastnost „client:load“, která zajistí, že uživatel bude mít možnost interakce s aplikací. V opačném případě by se v prohlížeči vypsala nefunkční komponenta. Celá komponenta musí být ohraničena párovým tagem , <main></main> nebo prázdným <></>. Komponenta tlačítka v hlavním souboru „index.astro“ nemusí být importována ani implementována, protože ji již obsahuje „Calculator.tsx“.

Importy případných závislostí a knihoven do komponent frameworku Astro se provádí na začátku souborů a jsou ohraničeny třemi pomlčkami.



Obrázek 28: Index.astro a implementace komponenty
Zdroj: vlastní tvorba

Díky této implementaci práce ukazuje reálné využití „island architecture“, která minimalizuje javascript na stanici uživatele a v případě nutnosti překresluje pouze tu komponentu, se kterou uživatel interagoval, a ne celou aplikaci.

2.5.4 Ukázka aplikace

Jelikož byla použita již nastylovaná a provozu schopná komponenta, tak vzhled i funkčnost je identický s aplikací v kapitole 2.4.4.

Aplikace je veřejně dostupná na adrese: http://bp-calc-astro.netlify.app.

2.6 Srovnání

Kapitola je zaměřena na porovnání frameworků, které byly použity v praktické části práce. Srovnávanými částmi jsou vývojářská přívětivost, výkon a základní uživatelská přívětivost.

2.6.1 Vývojářská přívětivost

Frontendové frameworky, jako je Astro či knihovna React, jsou z hlediska vývoje mnohonásobně lepší než programování webů pomocí klasických metod, jako je užití HTML, CSS a Vanilla JavaScriptu. Hlavními důvody jsou zlepšená struktura kódu, která umožňuje lépe udržovat a rozšiřovat projekt, větší efektivita při vytváření interaktivních a dynamických prvků a možnost využití komponentového přístupu. Komponenty usnadňují opakované použití kódu, což vede k větší modularitě a rychlejší údržbě. Kromě toho frameworky jako React a Astro často nabízejí optimalizované nástroje pro zlepšení výkonu a usnadnění vývoje díky široké komunitě a množství předpřipravených knihoven.

Samotné založení projektu je rychlé, intuitivní a první náhled funkční prázdné aplikace je reálný do pár minut. U běžného postupu je potřeba spousta kroků, jako je vytvoření několika souborů, které v sobě nesou strukturu webu, javascript a kaskádové styly. To vše je u frameworků automatizované.

React nebo Astro?

V konkrétní aplikaci kalkulačky jsem při vývoji v React knihovně využil populárnosti technologie a velmi širokou komunitu vývojářů, kde lze v případě potíží dohledat spoustu témat, případně nějaké otevřít. Možnost dohledání řešení problémů, nebo optimalizací pro kód staví React do popředí webových technologií.

Bohužel Astro nemá zatím tak rozsáhlou základnu, protože vzniklo teprve v roce 2019, tudíž má React náskok dlouhých šest let. To je v oblasti vývoje webů opravdu dlouhá doba.

Samotné založení projektu je proveditelné pomocí příkazu v terminálu, který nainstaluje holý projekt s potřebnými návaznostmi připravený ke spuštění. Je však potřeba dopředu promyslet, zda bude aplikace programována pomocí typescriptu, protože pro jeho snazší nastavení musí být zakládací příkaz doplněn o informaci, že si vývojář přeje nainstalovat typescript. Lze jej implementovat i dodatečně, avšak znamená to nutnost použití dalšího příkazu a úpravy několika souborů.

Na poli založení projektu vede Astro, které programátora provede instalací s dotazy, zda chce použít typescript, či ne.

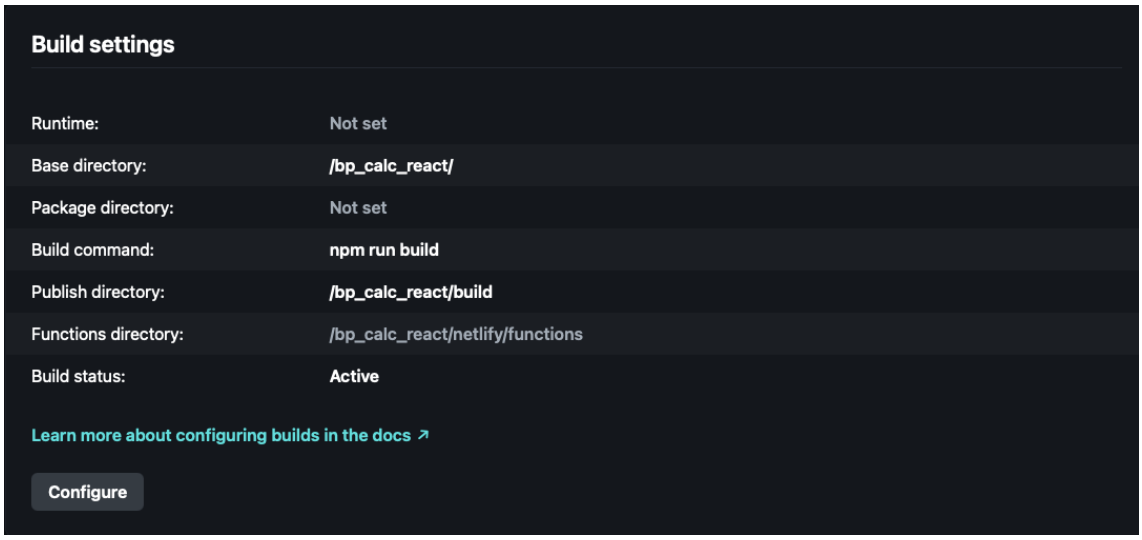
Co se psaní kódu týká, React je velmi intuitivní a snadný pro základní porozumění. Stačí znát typescript nebo javascript a případně umět číst v dokumentaci, kterou má knihovna velmi kvalitně zpracovanou.

Při implementaci v Astru byla ukázkově demonstrována udržitelnost a efektivita používání frameworků. Použití již vyvinutých komponent způsobilo, že vývoj kalkulačky trval velice krátkou dobu, tudíž je efektivita vývoje velmi vysoká.

Při programování jak v Reactu, tak Astru, jsem nenarazil na žádnou překážku, která by zpomalila vývoj.

Za zmínku v kapitole o vývoji aplikace stojí také její nasazení, které je plně automatizované díky verzovací platformě GitHub a online cloudové službě Netlify, která umožňuje snadné nasazení, spravování a hostování webové aplikace. Po prvotním propojení účtů a základním nastavením

nasazení se již každá nová úprava aplikace automaticky projeví po odeslání změny na platformu GitHub. Automatizace nasazení šetří čas a zajišťuje, že aplikace je vždy aktuální, což přispívá k lepší údržbě projektu.



Obrázek 29: Ukázka konfigurace na platformě Netlify
Zdroj: vlastní tvorba

2.6.2 Výkon

Výkon obou aplikací se měří pomocí nástroje „Lighthouse“, který je integrovaný do prohlížeče Google Chrome a slouží k analyzování výkonu a kvality stránek. Vývojářům nabízí podrobné informace, jak upravit web či aplikaci pro lepší uživatelský zážitek, výkon a marketingovou optimalizaci.

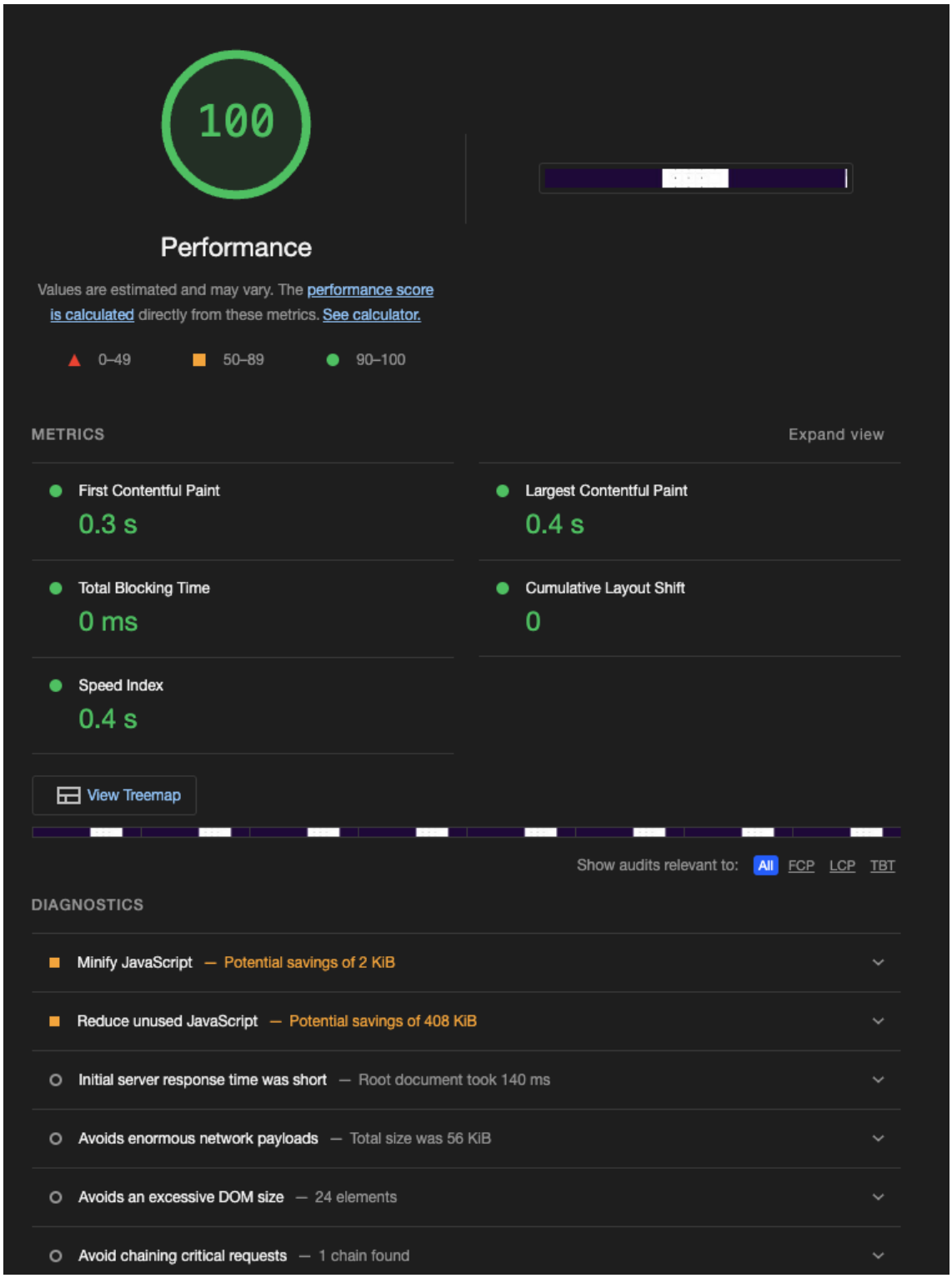
V rámci zmíněného nástroje pro měření se zaměříme na položku výkon, přestože výsledek máme například i pro SEO či přístupnost webu, protože se nejedná o cíle této práce.

Měření proběhlo u obou aplikací dvakrát, jednou pro desktop a jednou pro mobilní zařízení.

Výkon nám ukáže čtyři hodnoty:

- First Contentful Paint – První okamžik, kdy uživatel uvidí nějaký obsah (obrázek, tlačítko). Je důležité mít první vizuální zobrazení co nejrychlejší.
- Largest Contentful Paint – Čas vykreslení největšího viditelného prvku na obrazovce.
- Speed Index – Měří, jak rychle se obsah stránky zobrazuje vizuálně uživateli.
- Total Blocking Time – Doba, kdy stránka nereagovala na uživatelské interakce (posouvání, kliknutí).

U všech těchto hodnot platí, že čím je nižší, tím je aplikace výkonnější.

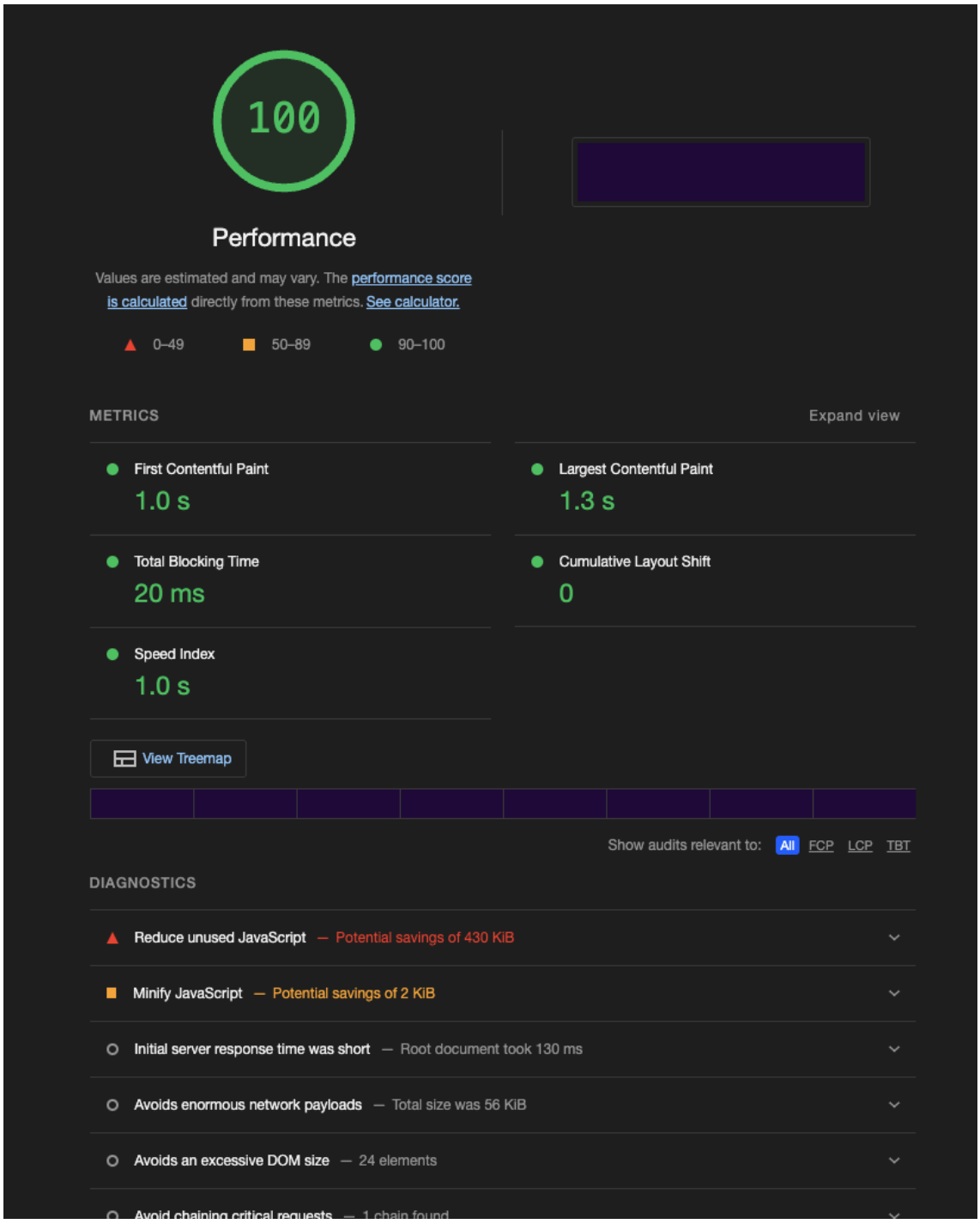


Obrázek 30: Výkon React – desktop

Zdroj: vlastní tvorba

React – desktop

- First Contentful Paint – 0,3 s.
- Largest Contentful Paint – 0,4 s.
- Speed Index – 0,4 s.
- Total Blocking Time – 0 ms.

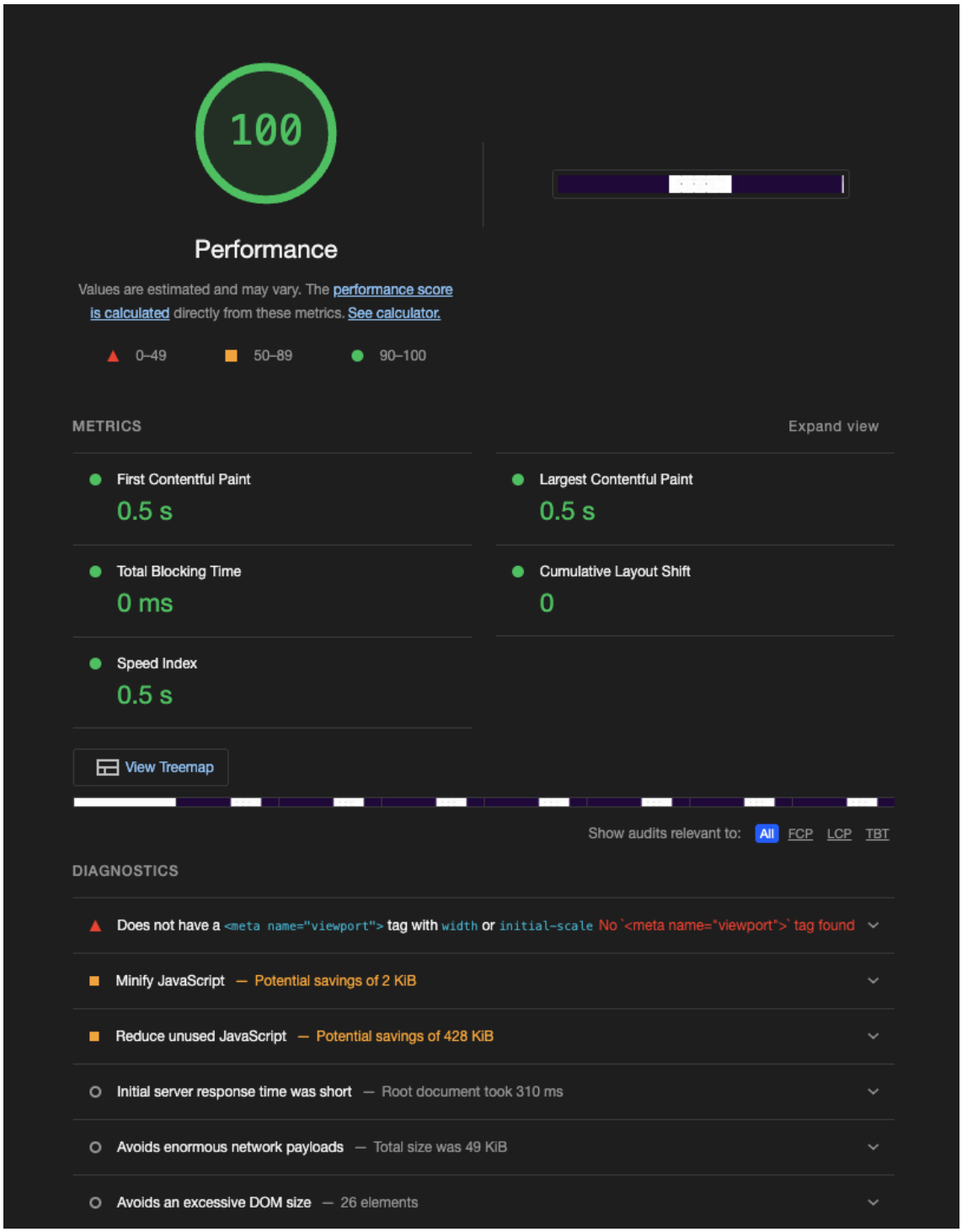


Obrázek 31: Výkon React – mobilní zařízení

Zdroj: vlastní tvorba

React – mobilní zařízení

- First Contentful Paint – 1 s.
- Largest Contentful Paint – 1,3 s.
- Speed Index – 1 s.
- Total Blocking Time - 20 ms.

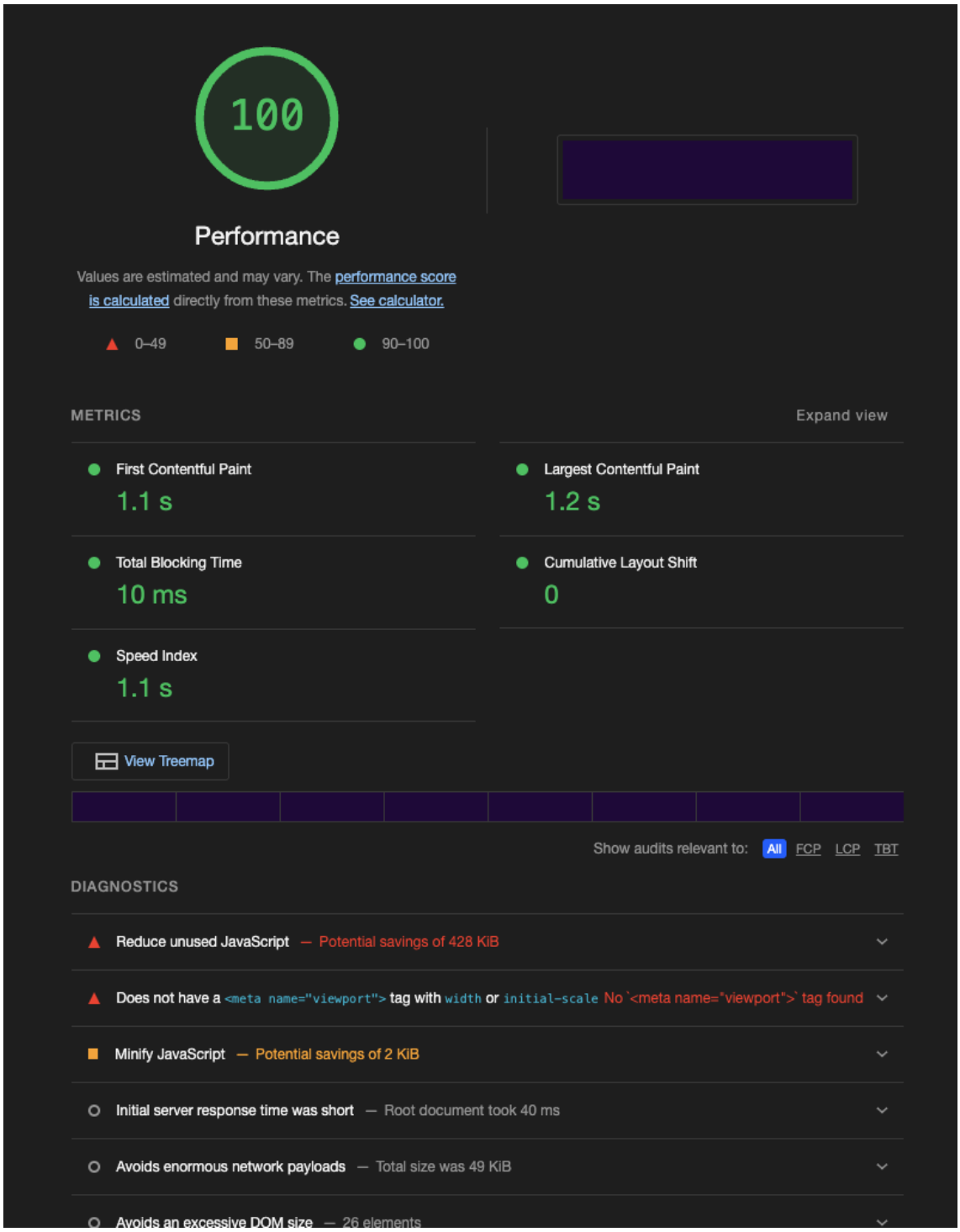


Obrázek 32: Výkon Astro – desktop

Zdroj: vlastní tvorba

Astro – desktop

- First Contentful Paint – 0,5 s.
- Largest Contentful Paint – 0,5 s.
- Speed Index – 0,5 s.
- Total Blocking Time – 0 ms.



Obrázek 33: Výkon Astro – mobilní zařízení

Zdroj: vlastní tvorba

Astro – mobilní zařízení

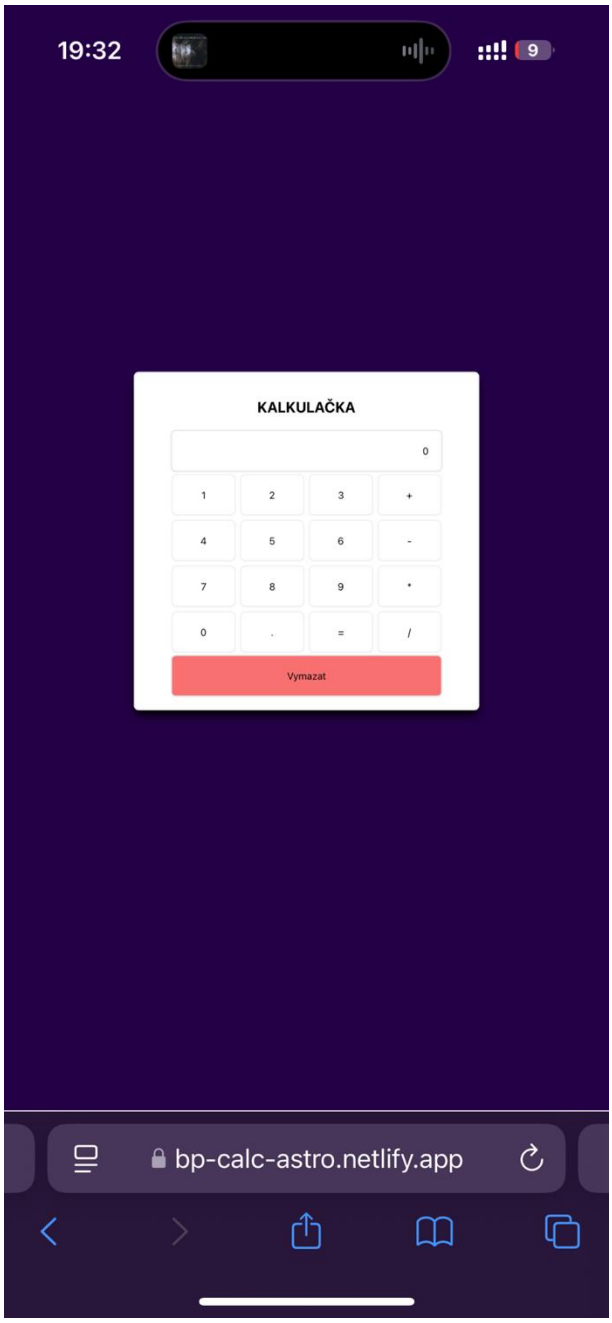
- First Contentful Paint – 1,1 s.
- Largest Contentful Paint – 1,2 s.
- Speed Index – 1,1 s.
- Total Blocking Time – 10 ms.

Výsledky všech měření výkonu byly stoprocentní, protože naměřené hodnoty byly nízké, až zanedbatelné.

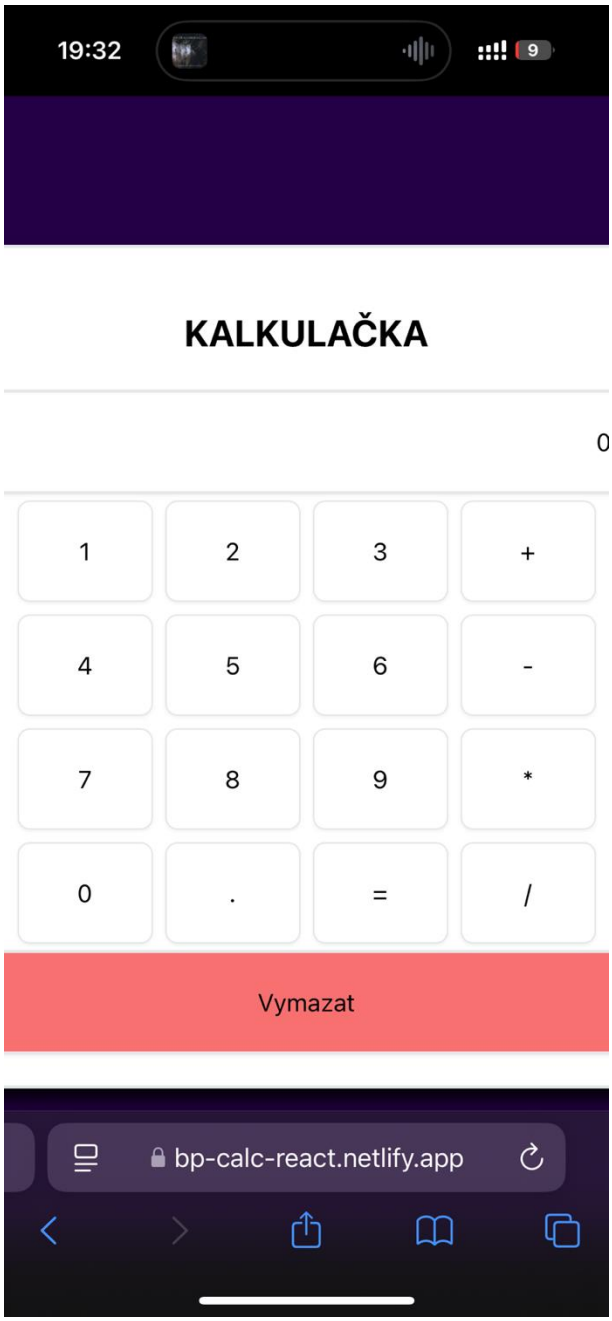
2.6.3 Uživatelská přívětivost

Z důvodu, že funkčně i vzhledově jsou obě aplikace identické, tak desktopové uživatelské rozhraní nelze srovnávat.

Kde je však výrazný rozdíl, je mobilní zobrazení. Při vývoji nebyl brán zřetel na responzivitu aplikace a verze implementovaná ve frameworku Astro byla příliš malá, a naopak kalkulačka React knihovny roztažená mírně přes okraj displeje. Funkčnost však byla v pořádku, na rozdíl od Astro aplikace, kde byla výrazně větší prodleva po stisku tlačítka, přestože při měření výkonu vyšlo sto bodů ze sta možných.



Obrázek 34: Mobilní zařízení – Astro
Zdroj: vlastní tvorba



Obrázek 35: Mobilní Zařízení – React
Zdroj: vlastní tvorba

2.6.1 Shrnutí

Kritérium porovnání	React	Astro	Komentář
Založení projektu	Pomocí příkazu, Typescript požaduje specifikaci zvlášť	Intuitivní nastavení po spuštění příkazu	Astro nabízí přívětivější založení projektu
Komunita	Široká komunita	Malá, rozvíjející se komunita	React má mnohonásobně větší uživatelskou základnu
Programátorská přívětivost	Opora rozsáhlé dokumentaci	Opora rozsáhlé dokumentaci	Obě technologie mají dobře zpracovanou dokumentaci, vývojářská přívětivost je tedy na vysoké úrovni
Nasazení	Snadné a automatizované nasazení	Stejná jednoduchost nasazení	Lze lehce vytvořit automatizované nasazení aplikací
Výkon na desktopu	Vysoký	Vysoký	Rozdíly by byly znatelné u rozsáhlejší aplikace
Výkon na mobilním zařízení	Vysoký	Pomalá odezva na interakce	Rychlost Astro aplikace byla výrazně pomalejší
Komponenty	Možnost recyklace komponent	Využití komponent jiných frameworků či knihoven	React omezen pouze na vlastní komponenty

Tabulka 1: Shrnutí srovnání

Zdroj: vlastní tvorba

Závěr

V bakalářské práci bylo provedeno prozkoumání a porovnání několika moderních frontendových frameworků, které lze vybrat vývoji optimalizovaných, dynamických webů a webových aplikací. Vue.js, Astro a knihovna React byla doplněna CSS frameworkem Tailwind, který do oblasti stylů přináší mnoho možností.

React a Vue.js poskytují robustní systémy pro správu komponent a stavu, což zjednodušuje vývoj interaktivních a dynamických uživatelských rozhraní. Astro přináší nový pohled na vývoj frontendu pomocí „island architecture“ a minimalizace klientem spouštěného JavaScriptu, což vede k výkonnějším a efektivnějším webovým stránkám.

Při výběru frameworku je nejdůležitější si uvědomit, jaká aplikace nebo stránka je vyvíjena. Pokud se jedná o komplexní aplikaci, je vhodné použít React. Pokud se však jedná o méně dynamickou aplikaci, kde je potřeba velký výkon a dokonalá optimalizace, jednoznačným favoritem je Astro.

Práce poskytuje pohled na současnou situaci na poli frontendových technologií, jejichž vývoj je neuvěřitelně rychlý. Proto, pokud se chce člověk tomuto oboru věnovat, je důležité neustálé vzdělávání a rozšiřování znalostí.

V rámci praktické části byla vyvořená jednoduchá aplikace kalkulačky v Reactu a v Astru, ve kterém po použití komponenty Reactu byla ukázána efektivnost a udržitelnost používání frontendových frameworků.

Porovnání vývojářské náročnosti neukázalo výrazné rozdíly v implementaci či spuštění. Odlišnost nebyla znatelná ani po použití nástroje měření výkonu.

Výsledek měření mě jako autora aplikací překvapil, a to především stoprocentními výsledky u obou verzí. Mým předpokladem bylo, že aplikace psaná v Astro bude výkonnější než v Reactu. Příčinou tohoto nečekaného závěru je zřejmě velikost aplikace, při které není možné zaznamenat výrazné rozdíly v rychlosti a optimalizaci. Myslím, že kdyby se jednalo o rozsáhlejší aplikace, React by mohl mírně zaostávat z důvodu většího objemu javascriptu na straně uživatele.

Na mobilních zařízení příliš neobstála aplikace implementována v Astru, protože odezva stisku tlačítek byla znatelně delší než u Reactu.

Při výběru, v jakém frameworku pracovat je důležité vyhodnotit velikost aplikace a zda s aplikací bude uživatel interagovat. Pro méně dynamické weby či aplikace je určitě vhodnější Astro, protože na stanici uživatele nepřenáší javascript, tudíž bude prohlížení, například blogu, výrazně svižnější. Oproti tomu React je dobrou volbou pro interaktivní aplikace. Dle mého názoru je dokonalá možnost kombinace obou technologií a využití „island architecture“.

Mezi nejmodernějšími frameworky se řadí Svelte (vydán v roce 2016) a Solidjs (vydán v roce 2020), přesto bych všem začátečníkům poradil začít s knihovnou React, a to ze dvou hlavních důvodů. Prvním je široká komunita vývojářů, která zajišťuje dostupnost velkého množství materiálů a podpory. Druhým, neméně zajímavým důvodem, je vysoká poptávka firem po vývojářích ovládajících React.

Seznam použité literatury

ASTRO team. *Astro.build*. Online. 2023. Dostupné z: https://astro.build/. [cit. 2023-12-31].

FEHERVARI, Zoltan. *Most Popular Web Frameworks of the Year 2023: Top Insights*. Online. Bluebird International. 2007. Dostupné z: https://bluebirdinternational.com/most-popular-web-frameworks/#:~:text=Front%2DEnd%20Frameworks-,React,-Introduced%20by%20Meta. [cit. 2023-12-31].

GAWIN, Cole. *How Astro compares to Next.js for React apps*. Online. Logrocket. 2022. Dostupné z: https://blog.logrocket.com/how-astro-compares-next-js-react-apps/. [cit. 2023-12-31].

GitHub. Online. 2008. Dostupné z: https://github.com. [cit. 2024-11-07].

CHAUDHARY, Raj. *The Evolution of Frontend Frameworks: From Vanilla JS to Modern Component-Based Architectures*. Online. Medium. 2023. Dostupné z: https://medium.com/readytowork-org/the-evolution-of-frontend-frameworks-from-vanilla-js-to-modern-component-based-architectures-ea8ade52ffc1. [cit. 2023-12-31].

MICHÁLEK, Martin. *Tailwind CSS: další evoluční krok pro CSS frameworky*. Online. Vzhůru dolů. 2021. Dostupné z: https://www.vzhurudolu.cz/prirucka/tailwind-css. [cit. 2023-12-31].

MDN Web Docs. Online. 2024. Dostupné z: https://developer.mozilla.org. [cit. 2024-11-18].

Netlify. Online. 2014. Dostupné z: https://www.netlify.com. [cit. 2024-11-07].

React team. *React.dev*. Online. 2023. Dostupné z: https://react.dev/. [cit. 2023-12-31].

Stack Overflow. Online. 2008. Dostupné z: https://stackoverflow.com. [cit. 2024-11-07].

Tailwindcss. Online. 2023. Dostupné z: https://tailwindcss.com/. [cit. 2023-12-31].

Typescript. Online. 2012. Dostupné z: https://www.typescriptlang.org/. [cit. 2024-11-18].

Vue.js team. *Vuejs.org*. Online. 2023. Dostupné z: https://vuejs.org/. [cit. 2023-12-31].