Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Moderní vývoj a design webových aplikací s využitím frameworků

Bakalářská práce

Autor: Ondřej Honců
Studijní obor: Informační Management (IM3-P)

Vedoucí práce: Mgr. Hana Rohrová

Hradec Králové Květen 2024

Prohlášení:

Prohlašuji, že jsem bakalářskou zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 25. 04. 2024 Ondřej Honců

Poděkování:

Děkuji vedoucí bakalářské práce Mgr. Haně Rohrové za metodické vedení práce. Její podpora, rady a odborné znalosti byly pro mě neocenitelné.

Anotace

Tato práce nabízí vhled do propojení moderních přístupů ve vývoji webových aplikací s designem, kladoucí zvláštní důraz na jednoduchost, uživatelský zážitek a digitální inovace. Začíná analýzou charakteristik moderních webových aplikací, včetně responzivního designu a bezpečnostních aspektů, a pokračuje zkoumáním SPA (Single Page Application) frameworků, doplněným o konkrétní ukázky kódu, které demonstrují tvorbu uživatelsky orientovaného webového prostředí. Dále se zaměřuje na různé frameworky a jazyky vhodné pro vytváření intuitivních webových aplikací, a to prostřednictvím technik specifických pro každý z nich. Výsledkem je praktická ukázka moderní a intuitivní webové aplikace, která nabízí komplexní, avšak snadno pochopitelný pohled na problematiku vývoje webových aplikací. Tato práce je určena pro začínající tvůrce, jež si chtějí usnadnit svou práci, rozšířit své znalosti a úspěšně prosadit své aplikace na trhu.

Annotation

Modern web application development and design using frameworks.

This thesis offers insight into the integration of modern approaches in web application development with design, placing particular emphasis on simplicity, user experience and digital innovation. It begins with an analysis of the characteristics of modern web applications, including responsive design and security considerations, and continues with an exploration of SPA (Single Page Application) frameworks, complemented by specific code samples that demonstrate the creation of user-centric web environments. It then focuses on the various frameworks and languages suitable for creating intuitive web applications, through techniques specific to each. The result is a hands-on demonstration of a modern and intuitive web application that offers a comprehensive yet easy-to-understand view of web application development. This work is intended for aspiring developers who want to make their jobs easier, expand their knowledge, and successfully market their applications.

Obsah

Seznam obrázků

Seznam ukázek kódu

Cíl práce

Cílem této práce je poskytnout komplexní a praktické vhledy do tvorby webových aplikací a stránek pomocí moderních frameworků, s důrazem na framework React a jeho design webových aplikací. Teoretická část práce se zabývá představením základních principů, struktury a rozdílů mezi vybranými frameworky. Praktická část práce demonstruje aplikaci těchto teoretických znalostí na konkrétním modelovém příkladu webové aplikace vyvinuté ve frameworku React. Tímto způsobem má tato práce za cíl nejen poskytnout ucelený přehled o tvorbě webových aplikací v moderních frameworcích, ale také demonstrovat konkrétní postupy a techniky při využití Reactu pro design webových aplikací.

Tato práce nabízí užitečné zdroje pro široké spektrum uživatelů, především pro webové nadšence. Hlavním cílem je poskytnout inspiraci a praktické postupy pro tvorbu moderních webových aplikací, s důrazem na využití frameworků. Práce je uceleným průvodcem procesem tvorby webových aplikací. Zdůrazňuje základní principy a techniky, které lze použít při vytváření uživatelsky přívětivých a efektivních webových rozhraní. Ukazuje, jak využít moderní nástroje a technologie k dosažení vynikajícího designu a funkcionality.

Díky svému strukturovanému přístupu je také vhodná pro studenty a výzkumníky, jež se chtějí dozvědět více o tvorbě webových aplikací. Poskytuje jim konkrétní příklady a ukázky kódu, které mohou použít jako základ pro své vlastní projekty nebo výzkum.

Společnosti a podniky, které se zabývají vývojem webových aplikací, mohou tuto práci využít k inspiraci pro jejich projekty. Nabízí jim ucelený přehled o moderních postupech a trendech v oblasti webového designu a vývoje, což může přispět k lepšímu výkonu a konkurenceschopnosti jejich produktů na trhu.

1 Úvod

V digitálním věku, kdy se internet stává kritickým prvkem každodenního života, hraje moderní vývoj webových aplikací a webdesign nepostradatelnou roli. Tyto disciplíny nejsou pouze technickým procesem, ale spojují umělecké prvky s technickou expertizou, vytvářejíce tak vizuálně oslnivé a funkční webové stránky. V tomto kontextu je cílem práce důkladně prozkoumat a analyzovat klíčové aspekty webdesignu a vývoje webových aplikací. [1]

Moderní vývoj a návrh webových aplikací jsou postupy a metodiky, které se používají k vytváření a optimalizaci webových aplikací, aby byly efektivní, bezpečné a uživatelsky přívětivé. Tento proces zahrnuje několik klíčových aspektů. [2]

- User-Centric Design (Design zaměřený na uživatele). Moderní webové aplikace začínají tím, že se zaměřují na potřeby a očekávání uživatelů. Tím se zajišťuje, že aplikace bude uživatelsky přívětivá a splní požadavky svých uživatelů.
- Responsivní design. Vývojáři se zaměřují na vytvoření webového rozhraní, které bude dobře fungovat na různých zařízeních a obrazovkách, včetně mobilních telefonů, tabletů a desktopů.
- Bezpečnost. Bezpečnostní opatření jsou klíčovou součástí moderního vývoje webových aplikací. To zahrnuje ochranu proti útokům, šifrování dat a zabezpečení přístupu k citlivým informacím.
- Agilní vývoj. Moderní vývoj webových aplikací často využívá agilní metodiky, jako je Scrum nebo Kanban, aby byl proces vývoje flexibilní a adaptabilní.

Frameworky jsou nástroji při vývoji moderních webových aplikací, protože zrychlují proces vývoje a pomáhají vývojářům dodržovat osvědčené postupy.

Jaký je tedy hlavní význam frameworků?

1. Zrychlení vývoje. Frameworky poskytují hotové komponenty a nástroje, které umožňují vývojářům vytvářet webové aplikace rychleji. To šetří čas a zdroje.
2. Konzistence. Frameworky poskytují jednotnou strukturu a konvence, což zajišťuje konzistenci v kódu a usnadňuje spolupráci mezi vývojáři.
3. Bezpečnost. Mnoho frameworků obsahuje zabudované bezpečnostní mechanismy a ochrany proti běžným bezpečnostním hrozbám, což pomáhá ochránit webové aplikace před útoky.
4. Aktualizace a údržba. Frameworky obvykle poskytují mechanismy pro snadnou aktualizaci a údržbu aplikace, což je klíčové pro udržení aplikace aktuální a bezpečné.
5. Rozšiřitelnost. Mnoho frameworků umožňuje snadno přidávat další funkce a rozšíření do webové aplikace pomocí pluginů a rozšíření.

Vývojové frameworky usnadňují vývoj a návrh moderních webových aplikací tím, že poskytují nástroje a struktury, které umožňují vývojářům efektivněji a spolehlivěji vytvářet aplikace. Tím pádem mají vliv na různé oblasti, včetně marketingu, tím, že umožňují rychlejší a bezpečnější nasazení inovativních webových řešení a produktů na trh. [1][2][4]

V teoretické části je ukázána tvorba webových aplikací a stránek v moderních frameworcích s ukázkou kódu. Představení jejich základních principů, struktury a rozdílů mezi nimi. Následuje rozšíření aplikace ve vybraném frameworku na modelovém příkladu webové aplikace v praktické části.

2 Metodika zpracování

Během tvorby práce byla provedena analýza moderních frameworků zaměřených na tvorbu a design webových aplikací. Začátek práce zahrnoval představení základních principů, struktury a rozdílů mezi vybranými frameworky, což poskytlo hlubší povědomí o výhodách a funkcionalitách jednotlivých frameworků s ohledem na konkrétní použití.

Zkoumány byly charakteristiky moderních webových aplikací, které zahrnují různé aspekty, jako je interaktivita, responzivita, použití moderních technologií a trendy v designu a uživatelském rozhraní. V rámci analýzy byla také posouzena dostupnost moderních webových aplikací a možnosti využití různých služeb pro efektivní ukládání dat a správu infrastruktury. Úvod do webdesignu představil základní principy a trendy v oblasti designu webových stránek, včetně použití barev, typografie, layoutů a uživatelského rozhraní. Analyzovány byly jednostránkové frameworky, jako jsou Angular, React, Vue a Svelte, společně s dalšími pro vývoj webových stránek, jako jsou Express a Java.

Následně byla provedena tvorba webových aplikací a stránek v rámci vybraných frameworků. Každý framework byl aplikován na modelovém příkladu, což umožnilo praktické pochopení jejich funkcionalit a možností. Během tohoto procesu byly demonstrovány různé aspekty tvorby aplikací, včetně práce s komponentami, routováním, správou stavu a dalšími klíčovými koncepty.

Dalším krokem bylo provedení rozšíření webové aplikace ve vybraném frameworku React na základě modelového příkladu. Tento krok umožnil praktické uplatnění nových funkcí nebo konceptů webdesignu ve skutečném prostředí.

Práce slouží k bližšímu porozumění moderních frameworků a ukázání možností jejich praktického využití v procesu tvorby moderních webových aplikací.

3 Literární rešerše

Tato část se zaměřuje na vysvětlení základních konceptů spojených s vývojem webových aplikací a některých technologií využívaných pro tvorbu frontendových částí těchto aplikací. Nejdůležitější částí je zde pohled na moderní frameworky, jejich přednosti a výhody s hlubším pohledem na jejich tvorbu. Tato ukázka slouží jako nastínění a rozlišení jednotlivých frameworků. Principy vývoje a návrhu moderních webových aplikací poskytují klíčové směry pro tvorbu efektivních, uživatelsky přívětivých a spolehlivých aplikací. Důraz je kladen na porozumění potřebám uživatelů, efektivnost designu a zvýšení důvěryhodnosti aplikace. Kromě toho je poukázáno na strategie, jak zaujmout s webovou aplikací na trhu, což je důležitý aspekt pro úspěšné uplatnění aplikace v konkurenčním prostředí.

3.1 Principy vývoje a návrhu moderních webových aplikací

Principy vývoje a návrhu moderních webových aplikací představují klíčové směry, které určují strategie a postupy pro efektivní tvorbu uživatelsky přívětivých a spolehlivých aplikací. Tyto principy jsou založeny na důkladném porozumění potřebám uživatelů a aplikují se od samotného začátku procesu vývoje až po finální implementaci a udržování aplikace. [4][5][6]

Nejzásadnějším principem je stanovení priorit potřeb uživatelů. To vyžaduje systematický výzkum a sběr zpětné vazby od uživatelů, čímž jsou identifikovány klíčové funkce a prvky aplikace, které pro uživatele přinášejí největší hodnotu. Tento proces je často podporován metodikami jako jsou průzkumy, dotazníky, focus group a uživatelské testování. [5]

Minimalistický přístup k designu a uživatelskému rozhraní je prioritou, kde méně znamená více. Cílem je vytvořit aplikaci, která je intuitivní a snadno použitelná, aniž by uživatelé museli procházet složitými návody nebo instrukcemi. Zásadní je také princip známosti aplikace. Uživatelé by měli snadno rozpoznat a pochopit, jak aplikace funguje, a to díky konzistentnímu designu, navigaci, použití osvědčených vzorů a konvencí v uživatelském rozhraní. [3]

Oddělení zájmů je principem, který se zaměřuje na strukturování aplikace tak, aby jednotlivé části měly jasně definované úkoly a funkce. Tímto způsobem je zajištěna modularita aplikace a snadnější údržba a rozšíření v budoucnosti. Důvěryhodnost aplikace je aspektem, jenž zahrnuje zabezpečení dat, ochranu soukromí a transparentnost vůči uživatelům ohledně zpracování jejich osobních údajů. To je zvláště důležité v době, kdy se uživatelé stále více zajímají o ochranu svých soukromých dat. [5][6]

Posledním, avšak neméně důležitým principem, je intuitivní design. Tento princip klade důraz na snadnou a přirozenou interakci uživatelů s aplikací, kde je navigace logická a procesy jsou intuitivní a snadno pochopitelné bez nutnosti podrobných instrukcí. [5][6]

3.2 Jak zaujmout s webovou aplikací na trhu

V dnešní době je klíčové nejen vytvořit skvělou webovou aplikaci, ale také ji efektivně propagovat a zaujmout na konkurenčním trhu. Jednou z nejdůležitějších strategií je cílený výzkum trhu. Důkladné pochopení potřeb a preferencí cílové skupiny je zásadní pro úspěch aplikace. K tomu slouží analýza trhu, identifikace konkurence a zjištění, co dělají dobře a kde mají nedostatky. [7]

K úspěšnému zaujetí trhu je důležitá diferenciace. Musíte jasně definovat, co vaši webovou aplikaci odlišuje od ostatních na trhu. Může to být jedinečná funkce, vynikající uživatelské rozhraní nebo služby poskytované vaší aplikací. Diferenciace pomůže vytvořit silnou značku a zajistit, že si uživatelé zapamatují právě vaši aplikaci. [8][9]

Kromě toho je důležité vytvořit efektivní marketingovou strategii. Zvažte využití různých kanálů a nástrojů pro propagaci vaší aplikace. To může zahrnovat sociální média, obsahový marketing, placenou reklamu nebo PR kampaně. Důležité je také nezapomínat na optimalizaci pro vyhledávače (SEO), což pomůže zvýšit viditelnost vaší aplikace online.

Nesmíte zapomenout na udržování spokojenosti uživatelů, poskytovat vynikající uživatelskou zkušenost a aktivně naslouchat jejich zpětné vazbě. Loajalita uživatelů a pozitivní recenze mohou být pro aplikaci klíčové při budování pověsti a získávání nových uživatelů. S kombinací těchto strategií a důkladným plánováním je snazší efektivně zaujmout trh a dosáhnout úspěchu se svou webovou aplikací. [10][11][13]

3.3 Webová aplikace

Webové aplikace a webové stránky představují dva základní koncepty internetových technologií, avšak jejich povaha a účel se liší. Webové stránky jsou primárně pasivním zdrojem informací, který uživatelům poskytuje obsah ve formě textu, obrázků, videí a dalších médií. Tyto stránky jsou často statické a jejich obsah se nemění, pokud nejsou ručně aktualizovány administrátorem.

Na druhou stranu jsou webové aplikace aktivními prostředky interakce, které umožňují uživatelům provádět akce a manipulovat s obsahem. Tyto aplikace jsou dynamické a obsah se může měnit v závislosti na uživatelských interakcích. Webové aplikace jsou často navrženy tak, aby uspokojovaly určité potřeby uživatelů a umožňovaly jim vykonávat různé úkoly přímo v prohlížeči. To může zahrnovat přihlášení, registraci, vyhledávání, přidávání a mazání příspěvků. [31][32]

Webové aplikace využívají koncept klient-server architektury, kde klientem je obvykle webový prohlížeč a serverem je platforma, na které je aplikace provozována. Tato architektura umožňuje přístup k aplikaci prostřednictvím internetu a současně mnoha uživatelům najednou, což je zásadní pro mnoho webových aplikací, jako jsou sociální sítě či e-commerce platformy. Funkční mechanismus webových aplikací využívá protokolu HTTP pro komunikaci mezi klientem a serverem. Mimo klasický přístup k obsahu pomocí webových stránek může webový server fungovat také jako API (Application Programming Interface), poskytující data ve formátu snadno čitelném pro další aplikace. [32][33]

3.4 Databáze

Databáze jsou základním stavebním kamenem mnoha informačních systémů. Jedná se o organizační struktury, které slouží k ukládání a správě dat a umožňují nám k nim snadný přístup, manipulaci a vyhledávání. Pro lepší porozumění, databáze je jako virtuální úložiště, kde jsou data systematicky uspořádána do tabulek, sloupců a řádků. Jsou řízeny pomocí speciálních dotazovacích jazyků a programovacích rozhraní. Tyto jazyky umožňují uživatelům interakci s databází, provádění dotazů a manipulaci s daty. Mezi nejznámější jazyky pro ovládání relačních databází patří SQL (Structured Query Language) a jeho varianty. [34]

Struktura databáze

Základní stavební bloky databáze jsou tabulky, které reprezentují jednotlivé entity nebo objekty. Každá tabulka má svůj název a skládá se z řádků a sloupců. Každý řádek v tabulce představuje jeden záznam nebo položku dat. Sloupce určují typ dat, která jsou uložena v dané tabulce. Každý sloupec má svůj název a definuje typ dat, která může obsahovat (například text, čísla, datumy). Klíče jsou speciální sloupce v tabulkách, které jednoznačně identifikují každý záznam v tabulce. Primární klíč je unikátní identifikátor pro každý záznam v tabulce, zatímco cizí klíče slouží k navázání vztahů mezi různými tabulkami. [34][35]

Využití databází

Jedná se o organizační struktury, které umožňují ukládání, správu a manipulaci s daty. Díky nim můžeme efektivně uchovávat a vyhledávat informace, analyzovat data a získávat cenné poznatky. Databáze jsou také nezbytnou součástí moderních webových aplikací, jež využívají dynamickou a interaktivní funkcionalitu. Vědecký výzkum a analýza dat využívají databáze k ukládání experimentálních dat a výsledků výzkumu. V logistice a distribuci pomáhají databáze sledovat zásoby, plánovat trasy a optimalizovat dodavatelské řetězce. V podnikovém prostředí jsou databáze často využívány pro správu obchodních operací, jako je účetnictví, správa zásob a fakturace. CRM systémy využívají databáze k uchování informací

o zákaznících a zlepšení vztahů se zákazníky. Analytické nástroje umožňují provádět rozsáhlé analýzy dat a vyhodnocovat trendy a chování zákazníků. [35]

3.5 Charakteristiky moderních webových aplikací

Moderní webové aplikace jsou výsledkem sofistikovaných digitálních nástrojů, vyvinutých na základě pokročilých technologických a designových principů, s cílem splnit rostoucí očekávání uživatelů a efektivně reagovat na dynamiku současných technologických trendů. Tyto aplikace se vyznačují několika klíčovými charakteristikami, jež podporují jejich úspěšné fungování a konkurenceschopnost na trhu digitálních technologií.

Jedním z primárních aspektů je vyšší očekávání uživatelů a nároky na výkon. Uživatelé dnes vyžadují nejen rychlé načítání a plynulou interakci, ale i atraktivní vizuální design a personalizovaný přístup. S narůstajícím povědomím o možnostech moderních technologií stoupá také jejich očekávání ohledně uživatelského zážitku. Webové aplikace proto musí být schopny promptně reagovat na uživatelské požadavky a poskytovat jim relevantní obsah a funkce. [3]

Klíčovou charakteristikou je dostupnost aplikací 24 hodin denně, 7 dní v týdnu z jakéhokoli místa na světě. To znamená, že aplikace musí být spolehlivě hostovány a disponovat robustní infrastrukturou, která umožní kontinuální provoz a minimalizaci výpadků. Uživatelé očekávají možnost přístupu k aplikaci kdykoli a odkudkoli, bez ohledu na geografickou polohu nebo časovou zónu.

Důležitá je použitelnost aplikace z různých zařízení a na různé velikosti obrazovek. Vzhledem k různorodosti zařízení, na kterých uživatelé přistupují k internetu, musí být webové aplikace responzivní a přizpůsobitelné. To znamená, že musí bezchybně fungovat na mobilních telefonech, tabletech i desktopových počítačích, a to bez ohledu na velikost obrazovky či rozlišení. Bezpečnost je klíčovým faktorem, jenž moderní webové aplikace musí garantovat. S ohledem na citlivost a objem dat, které tyto aplikace zpracovávají, je nezbytné zajistit ochranu těchto dat a prevenci před různými typy útoků. Aplikace musí být navrženy s ohledem na

bezpečnostní prvky a postupy, aby ochránily uživatelské údaje a udržely důvěru uživatelů. [3][4]

Moderní webové aplikace často využívají bohaté uživatelské rozhraní. To je zpracováno za pomoci JavaScriptu webovým prohlížečem na straně klienta, za častého využití moderních frontendových frameworků. Tyto aplikace efektivně komunikují s backendem prostřednictvím webových rozhraní jako např. REST api, což umožňuje asynchronní interakci a zlepšuje celkový výkon aplikace. [4]

Strategie a techniky

Personalizace obsahu je prvkem moderních webových aplikací. Jednou z efektivních strategií je využití algoritmů strojového učení k analýze chování uživatelů a následnému poskytování personalizovaných doporučení či obsahu. Příkladem může být personalizovaná nabídka produktů na základě historie prohlížení nebo nákupních preferencí uživatele.

Pro dosažení lepšího uživatelského zážitku mohou být využity interaktivní prvky jako animace, gesta a mikrointerakce. Tyto prvky přidávají do aplikace dynamiku a zvyšují angažovanost uživatelů. Důležité je však dbát na to, aby tyto prvky nepoškodily uživatelskou zkušenost, a proto je třeba je používat zdrženlivě a v souladu s potřebami uživatelů.

Jednoduchá a intuitivní navigace udržuje uživatele na stránce. Snadná orientace zaručuje, že uživatelé rychle a bezproblémově najdou požadovaný obsah. Důležité je poskytnutí možnosti aktivní interakce s aplikací. Uživatelé by měli mít široké možnosti komunikace, ať už jde o vyplňování formulářů nebo sdílení svých názorů. Pro dosažení tohoto cíle lze využít například konceptu „flat designu“, který minimalizuje složitost navigace a zlepšuje orientaci uživatelů na stránce. Průběžná analýza uživatelského chování a pravidelné monitorování poskytují cenné informace pro další zlepšení aplikace. Pro efektivní sběr a analýzu dat lze využít nástroje jako Google Analytics, které umožňují sledování interakcí uživatelů a identifikaci oblastí, jež je třeba vylepšit. [3]

3.6 Dostupnost moderních webových aplikací

Moderní webové aplikace jsou nedílnou součástí digitálního světa a zajištění jejich nepřetržité dostupnosti je zásadní pro uspokojení rostoucích očekávání uživatelů. Tohoto cíle lze dosáhnout prostřednictvím pečlivě navržených opatření v oblasti hostování a správy infrastruktury. [36]

Hlavní přístup k zajištění dostupnosti moderních webových aplikací jsou cloudová řešení. Využití cloudových platforem jako Amazon Web Services (AWS), Microsoft Azure nebo Google Cloud Platform umožňuje škálovatelné a flexibilní řešení pro hosting webových aplikací. Cloudové služby poskytují vysoký výkon a škálovatelnost, což zajišťuje udržení dostupnosti aplikací v různých situacích.

Důležitým faktorem je geograficky rozložená infrastruktura. Distribuce infrastruktury na různá geografická místa, známá také jako geografická replikace, pomáhá minimalizovat riziko výpadků v případě problémů v jednom regionu. Tímto způsobem se zajišťuje rovnoměrná dostupnost pro uživatele po celém světě.

Pro bezproblémový provoz je nutná efektivní správa infrastruktury. Automatizace procesů, jako je automatizace nasazování a správy aplikací, minimalizuje lidské chyby a zvyšuje efektivitu provozních procesů. To umožňuje rychlé reakce na případné problémy a udržení dostupnosti aplikací v optimálním stavu. Systémový monitoring a analýza dat slouží pro identifikaci potenciálních rizik a rychlou diagnostiku problémů. Kontinuální sledování výkonu a dostupnosti umožňuje okamžitou reakci na nežádoucí události a minimalizuje dopady výpadků.

Implementace bezpečnostních opatření, jako je šifrování dat, firewall a pravidelné aktualizace, snižuje riziko kybernetických útoků a chrání aplikace i uživatele před potenciálními hrozbami. Důležité je mít připravený plán pro případ výpadku. Vytvoření takového plánu umožňuje rychlé a koordinované akce v případě neočekávaných událostí a minimalizuje dopady výpadků na uživatele a podnikání. [36][37]

3.7 Úvod do webdesignu

Webdesign je umění a technika vytváření vizuálně atraktivních a funkčních webových stránek. Jedná se o proces, při kterém designéři kombinují estetické prvky, jako jsou barvy, typografie, grafika a animace, s praktickými aspekty, jako je uživatelská přívětivost a navigace. Cílem webdesignu je nejen vytvořit vizuálně atraktivní weby, ale také vytvořit uživatelsky orientované prostředí, které usnadní návštěvníkům rychlé a efektivní interakce se stránkami. Moderní webdesign se také zabývá responsivním designem, aby se webové stránky mohly přizpůsobit různým zařízením a obrazovkám. Výsledkem kvalitního webdesignu je stránka, která jednak oslovuje vizuálně, jednak poskytuje optimální uživatelský zážitek a podporuje cíle stránky, ať už se jedná o informování, komunikaci nebo obchodování. [16]

Význam webdesignu v dnešní digitální době

V dnešní digitální éře, kdy internet hraje klíčovou roli v komunikaci, obchodování a sdílení informací, má webdesign zásadní význam. Webové stránky jsou často prvním kontaktem mezi firmou nebo organizací a jejími zákazníky. Dobře navržený web může vyvolat pozitivní dojem, zvýšit důvěryhodnost a angažovanost uživatelů. [16][17]

Role webdesignu při vytváření stránek

Role webdesignu je kritická při vytváření úspěšných webových stránek. Design ovlivňuje uživatelský zážitek, navigaci, vizuální identitu a schopnost stránek komunikovat s cílovou skupinou. Správně navržený webdesign zohledňuje uživatelské potřeby, zpřístupňuje informace a usnadňuje interakci. Responzivní design zajišťuje optimální zobrazení na různých zařízeních, což je klíčové pro udržení návštěvníků na stránkách. Lze říct, že webdesign má významný dopad na celkový úspěch webových stránek a hraje roli prostředníka mezi obsahem a uživatelem, což je pro dosažení cílů a efektivitu webových prostorů to nejdůležitější. [12][16][17]

3.8 Jednostránkové aplikace (SPA)

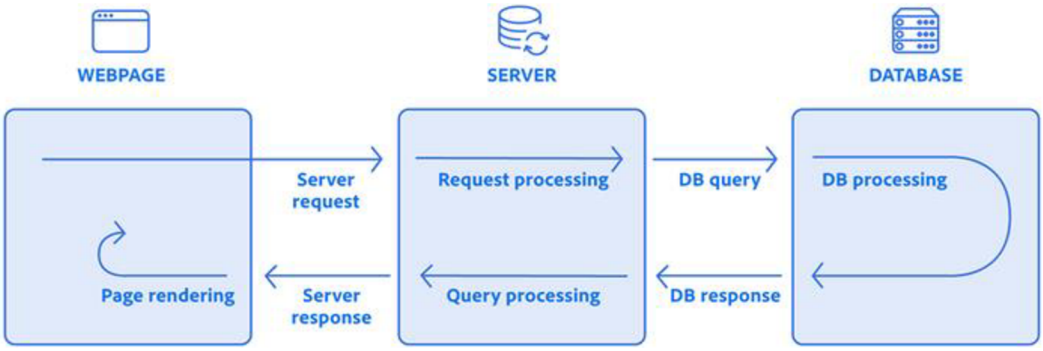
Jednostránkové aplikace (SPA – Single-Page Application) jsou moderní webové aplikace, které jsou navrženy tak, aby poskytovaly uživatelům plynulý a interaktivní zážitek. Při tvorbě SPA je kladen důraz na optimalizaci výkonu a zlepšení uživatelského zážitku. V rámci SPA je obsah načítán dynamicky do jediné stránky, což umožňuje snížit latenci a zkrátit dobu načítání stránek. [21]

Výhody SPA spočívají v rychlosti, efektivitě a plynulosti uživatelského zážitku. SPA minimalizují potřebu přenačítávání celých stránek při každé interakci uživatele, což vede k rychlejší odezvě aplikace a zlepšuje uživatelský zážitek. Díky použití technologií jako JavaScript, AJAX a moderních frameworků jako React, Angular nebo Vue.js jsou SPA schopny poskytnout dynamický obsah a interaktivní prvky bez nutnosti přenačítání stránky. [38][39]

Další výhodou SPA je offline použití, které umožňuje uživatelům používat aplikaci i bez připojení k internetu. SPA mohou ukládat data offline a synchronizovat je se serverem, když je připojení k internetu obnoveno. Tato funkcionalita je užitečná pro uživatele v oblastech s nepravidelným nebo slabým signálem internetu.

Škálovatelnost a údržba jsou dalšími aspekty SPA. Díky oddělení backendu a frontendu jsou SPA snadněji škálovatelné a umožňují nezávislé škálování obou částí aplikace. Jednodušší architektura SPA může vést ke snížení komplexity kódu a zjednodušení údržby aplikace.

SPA představují moderní a efektivní přístup k tvorbě webových aplikací, který nabízí výhody v rychlosti, efektivitě, škálovatelnosti a uživatelském zážitku. Jejich využití je vhodné pro projekty, které kladou důraz na plynulý a interaktivní zážitek uživatele. [38][39]



Obrázek 1 Jak fungují SPA
Zdroj: Adobe Experience Cloud Blog [39]

Klasická webová aplikace funguje tak, že když uživatel navštíví stránku, prohlížeč požádá server o kompletní HTML dokument obsahující strukturu, obsah a styly dané stránky. Poté, co server odešle tento dokument, prohlížeč ho zpracuje a zobrazí uživateli. Pokud uživatel provede nějakou interakci (například kliknutí na odkaz), celý proces se opakuje, tzn. prohlížeč znovu pošle požadavek na server s danou stránkou, server požadavek zpracuje a vrátí nový HTML dokument, který prohlížeč zpracuje a zobrazí uživateli. Na druhé straně Single Page Application (SPA) funguje jinak. Když uživatel navštíví stránku, prohlížeč načte pouze jeden HTML dokument, který obsahuje kostru aplikace, včetně JavaScriptových a CSS souborů. JavaScript pak zahrnuje veškerou logiku a šablonování aplikace. Jakmile je tento dokument načten, veškerá interakce uživatele s aplikací probíhá dynamicky na straně klienta. SPA používá asynchronní volání k načítání pouze potřebných datových částí zpětně ze serveru, což umožňuje rychlý a plynulý uživatelský zážitek bez nutnosti načítání nových stránek ze serveru při každé interakci. [38][39]

Stav aplikace

SPA udržují stav aplikace na straně klienta, což znamená, že při změně obsahu nebo navigaci se nemusí uživatelé znovu přihlašovat nebo ztrácet své aktuální místo v aplikaci.

SPA jsou ideální pro interaktivní a komplexní webové aplikace, které vyžadují plynulý a rychlý uživatelský zážitek. Jsou často používány v kombinaci

s moderními technologiemi, jako jsou API, a umožňují vývojářům vytvářet sofistikované webové aplikace s minimální režií na straně serveru.

Vývoj single-page webových aplikací zprostředkovávají javascriptové frameworky a knihovny, jako například. [19]

- Angular
- React
- Vue.js
- Svelte

3.9 Node.js

Node.js přináší revoluci v oblasti vývoje serverových aplikací. Jedním z klíčových rozdílů oproti tradičním aplikačním serverům je jeho asynchronní a událostmi řízený model. Na rozdíl od klasických serverů, které vytvářejí nové vlákno pro každý požadavek, Node.js efektivně využívá jediné vlákno, což umožňuje obsluhovat více požadavků současně. Tím se minimalizuje zátěž na systémové prostředky a zvyšuje se odezva aplikace. [2][4][40][41]

Prvkem Node.js je použití jazyka JavaScript na straně serveru. Tato jednotnost jazyka usnadňuje vývojářům pracovat se stejným jazykem na obou stranách, jak klienta, tak i serveru. Node.js je známý svou jednoduchostí a rychlostí vývoje, což usnadňuje tvorbu agilních a efektivních aplikací. V porovnání s tradičními aplikačními servery přináší Node.js také jednoduchost a škálovatelnost. S možností efektivně řešit problémy škálovatelnosti v reálném čase a s velkým počtem současných připojení je ideální pro moderní webové aplikace. [40]

Node.js má také rozsáhlou komunitu a ekosystém díky nástroji npm (Node Package Manager), který usnadňuje správu závislostí a integraci nových funkcionalit do projektu. V kombinaci s jednoduchostí, rychlostí a škálovatelností je Node.js atraktivní volbou pro vývoj moderních webových aplikací.

Node Package Manager

Node Package Manager (npm) představuje klíčový nástroj v ekosystému Node.js, zajišťující správu balíčků pro vývoj aplikací. Jeho funkcionalita usnadňuje procesy instalace, správy a sdílení JavaScriptových balíčků a jejich závislostí v rámci projektů. Vývojáři mohou rychle a jednoduše instalovat balíčky z oficiálního repozitáře npm pomocí příkazu "npm install <název-balíčku>". Tímto způsobem lze integrovat potřebné nástroje a knihovny do projektu. Pro inicializování souboru se použije příkaz npm init.

Ukázka vygenerovaného package.json

```
{
    "name": "moje-aplikace",
    "version": "1.0.0",
    "description": "Popis mé webové aplikace",
    "main": "aplikace.js",
    "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node aplikace.js"
  },
    "keywords": ["webová aplikace", "node.js", "express"],
    "author": "Jméno",
    "license": "ISC"
  }
```

Ukázka kódu 1 Package.json
Zdroj: Vygenerováno pomocí NPM

Další funkcí npm je správa závislostí. Balíček může mít definovány specifické závislosti na jiných balíčcích a npm se postará o sledování a aktualizaci těchto závislostí. To minimalizuje konflikty a zajišťuje konzistentní správu závislostí v projektu. Verzování balíčků je také důležitým prvkem. Každý balíček má svou vlastní verzi, což umožňuje vývojářům specifikovat požadovanou verzi balíčku v rámci jejich projektu. To přispívá k stabilitě a předvídatelnosti vývoje. Samotné publikování balíčků je snadné, což umožňuje vývojářům sdílet svůj kód s komunitou. Balíčky lze publikovat do centrálního repozitáře npm, a ostatní vývojáři je tak mohou jednoduše včlenit do svých projektů. [40][41]

Kromě toho npm umožňuje definovat vlastní scripty, které lze spouštět příkazem "npm run <název-skriptu>". Tato funkcionalita je užitečná pro automatizaci různých úkolů v rámci vývojového procesu. Npm vytváří robustní a efektivní prostředí pro správu balíčků a závislostí v Node.js projektech, což přispívá k plynulému a efektivnímu vývoji aplikací. [40][41]

3.10 Angular

Angular je moderní open-source framework vyvinutý společností Google, který slouží pro vývoj webových aplikací a jednostránkových aplikací (Single Page Applications – SPA). Jedná se o kompletní a komplexní nástrojovou sadu, která poskytuje strukturovaný a systematický přístup k vývoji aplikací pomocí jazyka TypeScript. [18]

Hlavními charakteristikami Angularu jsou komponenty, které představují základní stavební jednotky aplikace a jsou samostatnými, znovupoužitelnými a izolovanými bloky kódu. Každá komponenta obsahuje svou vlastní HTML šablonu, styly a logiku. Dále Angular obsahuje mnoho vestavěných direktiv, které umožňují manipulaci s DOM a přidávání dynamického chování do uživatelského rozhraní.

Důležitou vlastností je Dependency Injection (DI), což je mechanismus pro správu závislostí mezi různými komponentami a službami aplikace. Služby jsou v Angularu jednotkami pro sdílení funkcí a dat mezi různými částmi aplikace a jsou jednoduše vyměnitelné.

Angular.js a jeho architektura Model-View-Controller (MVC) nebo Model-View-ViewModel (MVVM) poskytují solidní rámec pro organizaci kódu aplikace a oddělení prezentační a aplikační logiky. Model reprezentuje data aplikace, View představuje uživatelské rozhraní a Controller/ViewModel řídí interakce mezi modelem a view, což napomáhá udržovat aplikaci strukturovanou a snadno spravovatelnou. [18][42]

Funkce Two-way Data Binding, která obsahuje Angular.js, znamená, že jakákoli změna v modelu aplikace automaticky ovlivní příslušné zobrazení v uživatelském rozhraní a naopak. Tato funkce zlepšuje interaktivitu aplikace a snižuje potřebu ruční synchronizace dat mezi modelem a view.

Directives a Pipes umožňují vývojářům manipulovat s DOM a transformovat data v uživatelském rozhraní. Directives umožňují přidávat vlastní chování do HTML, zatímco Pipes slouží k formátování dat v šablonách.
Angular.js podporuje modulární přístup k vývoji aplikací, což umožňuje vývojářům organizovat kód do samostatných a znovupoužitelných modulů. Tato modulární architektura zvyšuje přehlednost a snižuje redundanci kódu, což zlepšuje údržbu a škálovatelnost aplikace.

Angular také obsahuje vestavěný modul pro správu navigace v aplikaci pomocí definice tras (routes), což umožňuje dynamické načítání komponent podle URL adresy a správu stavu navigace. Dalšími funkcemi jsou například nástroje pro vytváření a správu formulářů, které zahrnují funkce pro validaci, sledování stavu formulářů a odesílání dat. [18][42]

Výhody angularu

Angular poskytuje robustní modul pro routování, který usnadňuje správu navigace v aplikaci a umožňuje vytvářet jednostránkové aplikace. Tímto způsobem můžeme definovat různé trasy a přiřadit jim odpovídající komponenty.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home.component';
import { AboutComponent } from './about.component';
```

Ukázka kódu 2 Angular, routování
Zdroj: Vlastní zpracování

Angular podporuje modularitu pomocí NgModule, to umožňuje oddělení funkcionalit do jednotlivých modulů, což zlepšuje přehlednost a udržitelnost kódu. Pomocí modulů můžeme seskupit související komponenty, služby a další zdroje do logických celků.

```
@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule ],
  declarations: [/* Deklarace komponent */],
  providers: [/* Deklarace služeb */],
  bootstrap: [/* Výchozí komponenty */]
})
export class AppModule { }
```

Ukázka kódu 3 Angular, NgModule
Zdroj: Vlastní zpracování

Angular využívá Dependency Injection pro správu závislostí mezi různými částmi aplikace. Tento přístup zlepšuje testovatelnost, modularitu a znovu-použitelnost kódu.

```
@Injectable()
export class LoggerService {
  log(message: string) {
    console.log(message);
  }
}
@Injectable()
export class UserService {
  constructor(private logger: LoggerService) {}
  getUserInfo() {
    this.logger.log('Fetching user information...');// Implementace
  }
}
```

Ukázka kódu 4 Angular, Injectable
Zdroj: Vlastní zpracování

3.11 React

React je populární open-source knihovna pro vývoj uživatelských rozhraní (UI), kterou vyvinula společnost Facebook (Meta). Jeho hlavním cílem je usnadnit vytváření efektivních a interaktivních webových aplikací s rychlým a plynulým uživatelským zážitkem. React se zaměřuje na deklarativní a efektivní psaní komponentů, jež mohou reagovat na změny stavu aplikace. React je často používán ve spojení s jinými nástroji a knihovnami, jako jsou React Router pro správu navigace nebo Redux pro efektivní správu stavu aplikace. Je vhodný pro vývoj jednostránkových aplikací (Single-Page Applications) a umožňuje vývojářům vytvářet moderní a výkonné webové aplikace. [43]

Komponenty

React komponenty jsou základními stavebními bloky v React.js, JavaScriptové knihovně pro tvorbu uživatelských rozhraní. Každá React komponenta je samostatným a znovupoužitelným prvkem, který obvykle reprezentuje určitou část uživatelského rozhraní. Tyto komponenty mohou být jednoduché, jako tlačítka nebo vstupní pole, nebo složitější, jako navigační panely nebo formuláře.

Každý React komponent může mít vlastní vstupy (props) a interní stav (state), které určují jeho chování a vzhled v závislosti na různých faktorech. Komponenty mohou být rovněž zanořeny uvnitř sebe, což umožňuje vytvářet hierarchii komponent a strukturovat kód aplikace.

Jedna z klíčových vlastností React komponent je možnost vykreslovat (renderovat) dynamický obsah na základě aktuálních dat a stavu aplikace. Tento přístup umožňuje jednoduše aktualizovat uživatelské rozhraní v reálném čase bez nutnosti ručně manipulovat s DOM (Document Object Model). [43]

Virtual DOM

Virtual DOM (VDOM) je koncept pro optimalizaci aktualizací uživatelského rozhraní. Jedná se o abstraktní reprezentaci aktuálního stavu DOM (Document

Object Model) ve formě stromové struktury, která je udržována v paměti. Tato virtuální reprezentace je mnohem lehčí a rychlejší na manipulaci než reálný DOM. Umožňuje optimalizovat proces aktualizací uživatelského rozhraní, jelikož minimalizuje počet operací na reálném DOM, a tím snižuje náročnost na výkon. Tento přístup vede ke zvýšení efektivity a rychlosti reakce aplikace na změny stavu. Při změně stavu aplikace v React.js jsou vytvářeny nové instance virtuálního DOM, které obsahují změny oproti předchozímu stavu. Následně React porovnává tyto virtuální stromy a identifikuje rozdíly mezi nimi. Na základě těchto rozdílů React generuje minimální sadu DOM operací potřebných k aktualizaci reálného DOM podle aktuálního stavu.

Jednosměrný tok dat

Jednosměrný tok dat je koncept, který React využívá k organizaci dat a interakce mezi komponentami v aplikaci. Tento přístup je založen na principu, že se data v aplikaci pohybují pouze jedním směrem, což usnadňuje správu stavu a zlepšuje předvídatelnost chování aplikace. V Reactu je jednosměrný tok dat implementován pomocí tzv. "props" (vlastnosti) a "state" (stav) komponent. Komponenty přijímají data prostřednictvím props, které jsou předávány z rodičovských komponent nebo z externích zdrojů a jsou pouze pro čtení. Tímto způsobem se zajišťuje, že data jsou předávána „shora dolů“ v hierarchii komponent. [43]

Stav (state) komponent je místní datová struktura, která určuje chování a vzhled komponenty. Tento stav je ovlivněn pouze uvnitř dané komponenty a nemůže být přímo modifikován zvenčí. Změny stavu jsou prováděny pomocí metody *setState(),* která zajistí aktualizaci stavu a znovu vykreslení komponenty. Tím, že data proudí jedním směrem od rodičovských k dceřiným komponentám a změny stavu jsou řízeny výhradně uvnitř samotných komponent, se minimalizuje složitost aplikace a zvyšuje se přehlednost a predikovatelnost chování. Jednosměrný tok dat je klíčovým konceptem v Reactu, který přispívá k jeho efektivitě a robustnosti. [43]

JSX

JSX (JavaScript XML) je rozšíření syntaxe JavaScriptu, které umožňuje psát HTML-like kód přímo v JavaScriptu. JSX je používán v Reactu pro psaní komponent, jež kombinují JavaScript a HTML-like syntaxi do jednoho souboru. Tímto způsobem je možné vytvářet uživatelská rozhraní pomocí deklarativní syntaxe, která je podobná psaní běžného HTML kódu. [43]

React Hooks

React Hooks jsou funkce, které umožňují komponentám React získávat stav, pracovat s efekty a provádět další operace. Dříve, než byly zavedeny Hooks, stav a další funkcionality byly dostupné pouze v třídových komponentách. S Hooks je možnost tyto funkce používat přímo ve funkčních komponentách, což usnadňuje správu stavu, efektů a dalších aspektů životního cyklu komponenty.

- *useState*: useState umožňuje komponentě React udržovat stav. Tento hook vrátí pár hodnot, aktuální hodnotu stavu a funkci, kterou můžete použít k aktualizaci stavu.
- *useEffect*: useEffect umožňuje provádět efekty v komponentě React. Například můžete použít useEffect k načtení dat z API nebo k odběru událostí. [43]

Importujeme React a useState z knihovny React. React se používá pro vytváření React komponentů a useState je jeden z „hooků“ (v tomto případě funkce), který umožňuje komponentám udržovat stav

```
// Funkční komponenta
const Counter = () => {
```

Ukázka kódu 5 React, funkční komponenta
Zdroj: Vlastní zpracování

```
// Příklad jednoduchého komponentu v React.js
import React, { useState } from 'react';
```

Ukázka kódu 6 React, jednoduchá komponenta
Zdroj: Vlastní zpracování

Definice funkční komponenty Counter. V Reactu můžeme vytvářet komponenty buď třídové, nebo funkční. V tomto případě je použita funkční komponenta.

```
const [count, setCount] = useState(0);
```

Ukázka kódu 7 React, hook pro sledování stavu
Zdroj: Vlastní zpracování

Pomocí useState hooku vytváříme proměnnou count a funkci setCount, která umožňuje aktualizovat stav. Výchozí hodnota stavu je nastavena na 0.

```
  const increment = () => {
    setCount(count + 1);
  };
```

Ukázka kódu 8 React, funkce pro zvýšení hodnoty count
Zdroj: Vlastní zpracování

Definice funkce increment, která zvýší hodnotu count o 1 po každém volání, poskytuje mechanismus pro dynamickou manipulaci s daty aplikace. Při kliknutí na tlačítko „Zvýšit“ se tato funkce zavolá, umožňující interaktivní aktualizaci stavu aplikace.

```
return (
    <div>
      <h1>Čítač: {count}</h1>
      <button onClick={increment}>Zvýšit</button>
    </div>
  );
};
```

Ukázka kódu 9 React, definice funkce increment
Zdroj: Vlastní zpracování

Návratová hodnota komponenty obsahuje JSX, který reprezentuje strukturu komponenty. Zobrazuje aktuální hodnotu count a tlačítko „Zvýšit“, které vyvolá funkci increment při kliknutí.

```
export default Counter;
```

Ukázka kódu 10 React, návratová hodnota
Zdroj: Vlastní zpracování

Komponent Counter je exportován, aby mohl být použit v jiných částech aplikace.

Toto je pouze základní ukázka čítače nebo také počítadla. React.js nabízí mnohem více funkcionalit pro efektivní vytváření uživatelských rozhraní. Komponenty, stav, životní cyklus, a mnoho dalších jsou základními koncepty v React.js.

3.12 Vue.js

Vue.js je moderní open-source JavaScriptový framework určený pro vývoj uživatelského rozhraní webových aplikací. Jeho základním cílem je usnadnit a zefektivnit proces tvorby interaktivních a dynamických webových stránek. Vue.js je navržen tak, aby byl snadno použitelný a zapadl do široké škály projektů, od malých jednostránkových aplikací až po rozsáhlé korporátní projekty. [44]

Fungování Vue.js je založeno na konceptu „progresivního frameworku“. To znamená, že je možné použít Vue.js postupně, postupně rozšiřovat jeho funkcionality a přizpůsobovat jej potřebám konkrétního projektu. Vue.js se zaměřuje na deklarativní přístup k vývoji uživatelského rozhraní, což znamená, že se programátoři zaměřují na definici toho, jak by mělo uživatelské rozhraní vypadat a jak by se mělo chovat, namísto toho, jak dosáhnout těchto vlastností pomocí manipulace s DOM (Document Object Model). [44]

Jedním z klíčových prvků Vue.js jsou komponenty. Komponenty jsou samostatné a znovupoužitelné bloky kódu, které obsahují šablonu (HTML kód), styly (CSS) a logiku (JavaScript). Komponenty umožňují strukturovat webovou aplikaci do jednotlivých částí, což usnadňuje správu a údržbu kódu. Další důležitou vlastností Vue.js je dvoucestný data binding. To znamená, že pokud dojde ke změně dat ve Vue.js komponentě, automaticky se tato změna promítne do uživatelského rozhraní a naopak. Tento mechanismus umožňuje synchronizaci dat mezi modelem (datová reprezentace aplikace) a pohledem (uživatelské rozhraní). Vue.js rovněž poskytuje řadu dalších funkcionalit, jako jsou direktivy (například v-bind pro vázání atributů HTML prvků na data Vue instance), událostní systém pro zachycování uživatelských akcí, systém routování pro správu navigace v aplikaci a mnoho dalšího. [44]

3.13 Svelte

Svelte.js je moderní JavaScriptový framework, který nabízí nový a inovativní přístup k vývoji webových aplikací. Byl vytvořen Richem Harrisem a poprvé představen veřejnosti v roce 2016. Od té doby získal na popularitě mezi vývojáři díky své jednoduchosti, efektivitě a schopnosti generovat čistý a optimalizovaný kód. [20]

Jedním z hlavních principů Svelte.js je koncept kompilace ve srovnání s interpretačním přístupem jiných frameworků, jako je například React nebo Angular. Svelte nevytváří virtuální DOM jako tyto frameworky, místo toho kompiluje komponenty do čistého JavaScriptu během build procesu. Tento přístup eliminuje nadbytečné vrstvy abstrakce a umožňuje generování kódu, který je efektivnější a rychlejší.

Svému jménu odpovídající, Svelte.js se vyznačuje minimalismem a jednoduchostí. Syntaxe frameworku je přehledná a intuitivní, což zjednodušuje proces učení a používání. Svelte se zaměřuje na to, aby vývojáři psali méně kódu a dosahovali přitom větší efektivity a čitelnosti.

Svelte.js obsahuje reaktivní model. Framework umožňuje vytvářet proměnné, které automaticky reagují na změny svých hodnot, což zjednodušuje sledování stavu aplikace a aktualizaci uživatelského rozhraní. Tento reaktivní model je založen na konceptu „reálných proměnných“ a eliminuje potřebu používat složité metody pro správu stavu aplikace.

Svelte.js rovněž podporuje komponentovou architekturu, jež umožňuje vývojářům rozdělit uživatelské rozhraní do samostatných a znovupoužitelných částí. Tato modulární struktura kódu zvyšuje přehlednost a usnadňuje údržbu aplikace v průběhu času. [20]

Klíčové koncepty Svelte.js

1. Inovativní Kompilace, Svelte.js přináší nový přístup k vytváření webových aplikací prostřednictvím kompilace komponent do čistého JavaScriptu. Tento inovativní přístup eliminuje potřebu virtuálního DOM a generuje kód, který je efektivnější a rychlejší.
2. Minimalismus a jednoduchost, Svelte.js se vyznačuje minimalistickým a přehledným přístupem, jenž zjednodušuje proces vývoje. Jeho syntax je intuitivní a umožňuje vývojářům psát méně kódu, aniž by to ovlivnilo výkon či funkcionalitu aplikace.
3. Reaktivní Model, framework podporuje reaktivní model, který umožňuje automatickou reakci na změny stavu aplikace. Tento koncept, založený na „reálných proměnných“, eliminuje složité metody pro správu stavu a usnadňuje aktualizaci uživatelského rozhraní.
4. Komponentová Architektura, Svelte.js přináší podporu pro komponentovou architekturu, jež umožňuje snadné rozdělení uživatelského rozhraní do samostatných a znovupoužitelných částí. Tato modulární struktura kódu zvyšuje přehlednost a usnadňuje údržbu aplikace.

   [20]

3.14 Další frameworky a jazyky pro vývoj webových stránek

Výběr nejlepšího frameworku pro vývoj webových stránek závisí na konkrétních potřebách, dovednostech a preferencích. Každý z frameworků má své vlastní výhody a je vhodný pro různé účely. [2][4]

Express

Express.js, framework pro Node.js, představuje klenot v nástrojové sadě pro vývojáře webových aplikací. Jedná se o minimalistický, avšak mocný framework, který výrazně zjednodušuje proces vytváření efektivních a škálovatelných serverových aplikací. [22]

Srdcem Express.js je jeho schopnost definovat a spravovat cesty (routy), což umožňuje mapování HTTP požadavků na určité části kódu nebo funkce. Tímto způsobem vývojáři mohou definovat, jak bude server reagovat na různé druhy požadavků na různých URL adresách.

Jedním z klíčových prvků Express.js je také jeho použití middleware. Middleware jsou funkce, jež mají přístup k objektu žádosti (request) a odpovědi (response). To umožňuje vývojářům provádět různé operace, například autentizaci, zpracování dat nebo kontrolu bezpečnosti, předtím, než je požadavek předán dalším částem aplikace. Framework podporuje širokou škálu šablonovacích enginů, což je klíčové pro generování dynamického obsahu. Tato flexibilita usnadňuje vývojářům vytváření esteticky příjemných uživatelských rozhraní.

Pro poskytování statických souborů, jako jsou obrázky, CSS nebo JavaScript, Express poskytuje vestavěný middleware. Tímto způsobem lze jednoduše obsluhovat veřejné soubory bez nutnosti složitých konfigurací. Express.js je také oblíbeným nástrojem pro vytváření RESTful API. Strukturovaný způsob, jakým Express zpracovává cesty a middleware, usnadňuje tvorbu rozhraní pro komunikaci mezi klientem a serverem. [22]

Ukázka frameworku Express.js

Nejprve je třeba nainstalovat Express.js pomocí nástroje npm (Node Package Manager). Po instalaci můžete vytvořit novou Express aplikaci pomocí express-generator nebo manuálně, stačí použít příkaz. [22]

```
npm install express –save
```

Express aplikace typicky obsahuje soubory, jako je app.js nebo index.js, které slouží jako vstupní bod aplikace. Dále můžete mít složky pro routy, kontroléry, šablony a další.

Middleware jsou funkce, jež mají přístup k objektu žádosti (request) a odpovědi (response). Jsou volány postupně podle pořadí, ve kterém jsou registrovány. Můžete vytvářet vlastní middleware nebo používat existující.

```
const express = require('express');
const app = express();

// Vlastní middleware
app.use((req, res, next) => {
  console.log('Middleware pro každý požadavek');
  next();
});
```

Ukázka kódu 11 Express, middleware
Zdroj: Vlastní zpracování

Routy definují, jak bude server reagovat na různé cesty (URL adresy). Každá cesta může mít přiřazené jedno nebo více middleware. Routy mohou být definovány modulárně v samostatných souborech.

```
// V jednoduchém případě
app.get('/', (req, res) => {
    res.send('Ahoj světe!');
  });
  // Využití modulárních rout
  const router = require('./routes/index');
  app.use('/api', router);
```

Ukázka kódu 12 Express, modulární routy
Zdroj: Vlastní zpracování

Express poskytuje mnoho vestavěných funkcí pro zpracování požadavků, manipulaci s cookies, práci s HTTPS, obsluhu šablon a mnoho dalšího. Zároveň umožňuje rozšíření funkcionalit pomocí různých balíčků a modulů. Pro generování dynamického obsahu můžete použít různé šablonovací enginy, jako například EJS nebo Pug. Pro poskytování statických souborů (CSS, obrázky) můžete využít vestavěný middleware. Express je oblíbeným nástrojem pro tvorbu RESTful API. Pomocí něj můžete jednoduše definovat cesty, middleware pro autentizaci a rychle vytvořit rozhraní pro komunikaci mezi klientem a serverem. [22]

Vytvoření jednoduché Express aplikace pro domovskou stránku

Domovská stránka byla vytvořena pomocí frameworku Express a Node.js. Pro vytvoření souboru index.js jsme nejprve vytvořili projektovou strukturu a inicializovali projekt pomocí příkazu *npm init*. Následně jsme nainstalovali Express framework příkazem *npm install express*.

```
// Import Express framework
const express = require('express');

// Vytvoření instance aplikace Express
const app = express();


// Definice cesty "/home" s ovladačem (handler) pro GET požadavky
app.get('/', (req, res) => {
    // Odeslání odpovědi klientovi s textem "Toto je domovská stránka"
    res.send('Toto je domovská stránka');

});
// Další zpracování cest by mohlo být přidáno zde

// Nastavení portu, na kterém server poslouchá
const port = 3000;

// Spuštění serveru a naslouchání na zadaném portu
app.listen(port, () => {
    // Vypsání informace o běžícím serveru do konzole
    console.log(`Server běží na http://localhost:${port}`);
});
```

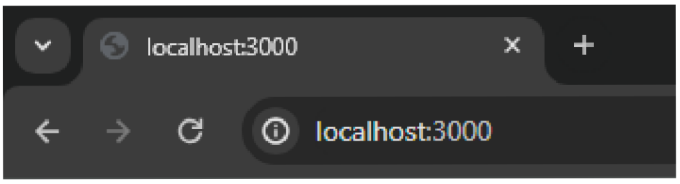
Ukázka kódu 13 Express, spuštění serveru
Zdroj: Vlastní zpracování

Index pak následně spustíme v terminálu příkazem node <název dokumetu.js >. Výstup z terminálu by měl pak vypadat takto:

```
PS C:\Users\Ondřej Honců\Visualko\Express> node server.js
Server běží na http://localhost:3000
```

Obrázek 2 Express, terminál
Zdroj: Vlastní zpracování pomocí terminálu

Abychom zobrazili domovskou stránku, musíme přejít ve webovém prohlížeči na danou adresu http://localhost:3000.



Toto je domovská stránka

Obrázek 3 Express, domovská stránka
Zdroj: Vlastní zpracování

Registrace uživatele

Tento kód vytváří jednoduchý server, který poskytuje registrační formulář na adrese http://localhost:3000 a zpracovává registraci na cestě /register. Samozřejmě, ve skutečné aplikaci by se měla bezpečně ukládat uživatelská jména a hesla, avšak tento kód slouží pouze k ilustraci základních principů.

```
const express = require('express');
const bodyParser = require('body-parser');
// Vytvoření instance aplikace Express
const app = express();
// Povolení používání bodyParser pro zpracování těla požadavků
app.use(bodyParser.urlencoded({ extended: true }));
const registrationForm = `
    <form action="/register" method="post">
        <label for="username">Uživatelské jméno:</label>
        <input type="text" id="username" name="username" required>
        <br>
        <label for="password">Heslo:</label>
        <input type="password" id="password" name="password" required>
        <br>
        <button type="submit">Registrovat</button>
    </form>
`;
// GET route pro zobrazení registračního formuláře
app.get('/', (req, res) => {
    res.send(registrationForm);
});
// POST route pro zpracování registračních dat
app.post('/register', (req, res) => {
    const username = req.body.username;
    const password = req.body.password;
    // Zde bychom měli provést skutečnou registraci, například uložit uživatele do
databáze, V tomto příkladu pouze vypíšeme údaje o registraci do konzole
    console.log(`Uživatel ${username} byl úspěšně zaregistrován s heslem
${password}`);
    res.send('Registrace byla úspěšná!');
});
// Nastavení portu, na kterém server poslouchá a Spuštění serveru a naslouchání
const port = 3000;
app.listen(port, () => {
    console.log(`Server běží na http://localhost:${port}`);
});
```

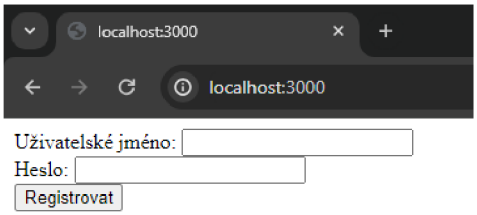
Ukázka kódu 14 Express, registrace uživatele
Zdroj: Vlastní zpracování

Opětovné zapnutí serveru pomocí příkazu node <jméno_souboru>.js.

```
PS C:\Users\Ondřej Honců\Visualko\Express> node server.js
Server běží na http://localhost:3000
```

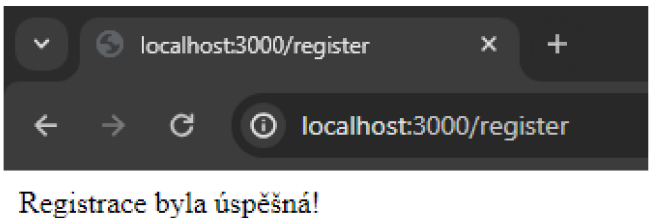
Obrázek 4 Express, výstup terminálu
Zdroj: Vlastní zpracování pomocí výstupu z terminálu

Po spuštění adresy http://localhost:3000 ve webovém prohlížeči se zobrazí registrační formulář.



Obrázek 5 Express, localhost 1/2
Zdroj: Vlastní zpracování

Po zpracováni registračních dat jsou uživatelé přesměrováni na adresu http://localhost:3000/register, kde je vypsán údaj „Registrace proběhla úspěšně“.



Obrázek 6 Express, localhost 2/2
Zdroj: Vlastní zpracování

Registrační údaje se zároveň vypisují do terminálu, samozřejmě pokud bychom chtěli provádět skutečné registrace, uživatele bychom měli ukládat do databáze.

```
PS C:\Users\Ondřej Honců\Visualko\Express> node server.js
Server běží na http://localhost:3000
Uživatel admin byl úspěšně zaregistrován s heslem admin
```

Obrázek 7 Express, výstup terminálu
Zdroj: Vlastní zpracování pomocí výstupu z terminálu

Výhody Express.js

Express.js je minimalistický, ale mocný framework pro tvorbu webových aplikací v Node.js. Jeho flexibilita a jednoduchost umožňují vývojářům vytvářet robustní a efektivní aplikace různého typu. Jednou z klíčových vlastností Express.js je jeho schopnost efektivně routovat a zpracovávat HTTP požadavky.

Routování v Express.js je základem pro definici cest a přiřazení jim odpovídajících ovladačů. Ovladače jsou funkce, které se vykonávají při zasílání požadavku na danou cestu. Tyto ovladače mohou provádět různé úkoly, jako je načtení dat z databáze, vygenerování obsahu, zpracování formulářů a další.

Middleware jsou další klíčovou součástí Express.js. Jsou to funkce, které se vykonávají před samotným zpracováním požadavku nebo po jeho vyřízení. Middleware může provádět různé úkoly, jako je ověřování uživatele, zpracování cookies, logování požadavků a další. Pomocí middleware můžeme jednoduše rozdělit logiku na menší a snadno spravovatelné části.

Express.js podporuje různé view enginy pro generování dynamických HTML stránek. EJS (Embedded JavaScript) a Pug (dříve známý jako Jade) jsou nejběžnější view enginy používané v Express.js. Tyto enginy umožňují vkládat dynamická data přímo do HTML šablon a usnadňují vytváření komplexních uživatelských rozhraní.

Pomocí Express.js můžeme také snadno servírovat statické soubory, jako jsou CSS, obrázky a JavaScript. Vestavěné middleware express.static umožňují definovat adresář veřejných souborů, které mají být poskytovány přímo klientům.

Pro vývoj RESTful API je Express.js také skvělou volbou. Pomocí různých metod, jako GET, POST, PUT a DELETE, můžeme definovat cesty pro různé operace. Express.js poskytuje jednoduché a intuitivní rozhraní pro vytváření a správu API. [22]

Java

Java je všeobecně používaný programovací jazyk a platforma pro vývoj a spouštění různých typů softwaru. Byla vyvinuta společností Sun Microsystems (nyní součástí Oracle Corporation) a poprvé představena v roce 1995. Java nabízí řadu vlastností, které ji činí vhodnou pro různé aplikace, od webových stránek po podnikové systémy. Vývoj v jazyce Java je rozsáhlý a pokrývá mnoho různých oblastí od vývoje webových a mobilních aplikací po podnikové a distribuované systémy. [46]

Vývoj v Javě má silnou podporu a rozsáhlou komunitu, což z ní činí oblíbenou volbu pro mnoho typů aplikací a projektů. Jednou z možností vývoje v jazyce Java jsou Java Server Pages (JSP), což je technologie zaměřená na dynamický vývoj webových stránek. Tato technologie se specializuje na implementaci prezentační vrstvy neboli GUI (grafické uživatelské rozhraní). Její princip fungování je srovnatelný s PHP, kde se Javovský kód vkládá přímo do HTML dokumentu. V současné době se však tato technologie příliš nevyužívá, a to zejména kvůli rostoucí popularitě moderních frameworků. [46][47]

Jedním z nejuznávanějších open-source frameworků je Spring. Tento framework poskytuje širokou škálu modulů, jako jsou autorizace, autentizace a komunikace s databází, které lze efektivně využít v závislosti na konkrétních požadavcích dané aplikace. Díky své flexibilitě a podpoře komplexních funkcí získal Spring velkou oblibu mezi vývojáři. [47]

Jednoduchý kód v jazyce Java pro „Ahoj světe!“.

```java
public class HelloWorld {
    // Definice třídy s názvem HelloWorld
    public static void main(String[] args) {
        // Vstupní metoda programu
        System.out.println("Hello, World!");
        // Vypíše "Hello, World!" do konzole
    }
}
```

Ukázka kódu 15 Java, „Ahoj světe!“
Zdroj: Vlastní zpracování

Samotný vývoj webových, mobilních i podnikových aplikací je založen na pevném základu znalostí a vhodných nástrojích. V oblasti webových aplikací je využívána technologie jako JavaServer Faces (JSF), která poskytuje robustní komponentní model pro snadnou tvorbu uživatelských rozhraní. Dále je zde Spring Framework, jenž je využíván pro rozsáhlé enterprise aplikace. Je zaměřen na odstranění běžných problémů vývoje a zjednodušení konfigurace.

Při vývoji mobilních aplikací pro platformu Android se neobejdeme bez Javy, která tvoří základ pro tvorbu aplikací pro tuto platformu. Prostředí Android Studio je pak využíváno pro snadnou tvorbu aplikací optimalizovaných pro zařízení s Androidem.

Pro podnikové aplikace, jež vyžadují distribuovanou, transakční a bezpečnou architekturu, jsou využívány technologie jako Enterprise JavaBeans (EJB) a Spring Boot. Tyto nástroje jsou zaměřeny na usnadnění vytváření robustních a snadno udržovatelných podnikových aplikací s minimální konfigurací.

Databázový vývoj je klíčovým prvkem v mnoha typech aplikací. Standardní API Java Database Connectivity (JDBC) je využíváno pro efektivní komunikaci s relačními databázemi pomocí SQL dotazů. Framework jako Hibernate pak zjednodušuje práci s databází a mapování objektů na relační schéma.

Pro efektivní vývoj v Javě jsou nepostradatelné i vhodné vývojové prostředí. Eclipse a IntelliJ IDEA jsou dva z nejpopulárnějších nástrojů pro vývoj v Javě. Poskytují širokou škálu funkcí a nástrojů, které usnadňují práci a přispívají k efektivnímu vývoji aplikací. [46][47]

4 Tvorba aplikací a stránek za pomoci SPA

Tato kapitola je zaměřena na praktickou tvorbu aplikací a webových stránek s využitím vybraných frameworků, které byly představeny v předchozích kapitolách. Dosud bylo prozkoumáno mnoho single page frameworků a jejich základní principy. Nyní je čas na hlubší ponoření se do procesu vytváření aplikací s jejich pomocí.

Pro účely demonstrace bylo pracováno s frameworky, jež byly prozkoumány v předchozích částech této práce. Každý z těchto frameworků má své vlastní charakteristiky a výhody. Prostřednictvím konkrétních ukázek je ukázáno, jak lze s jejich pomocí efektivně vytvářet moderní webové aplikace.

Postup práce je prezentován krok za krokem, aby byl poskytnut jasný obraz toho, jak frameworky usnadňují proces vývoje aplikací. Jsou zahrnuty praktické příklady kódu, postupy a tipy pro efektivní využití těchto nástrojů.

Cílem praktické části je poskytnout užitečný vhled do procesu tvorby aplikací a stránek pomocí moderních frameworků. Po absolvování této části by mělo být lépe porozuměno konceptům a technikám spojeným s vývojem webových aplikací.

4.1 Angular aplikace

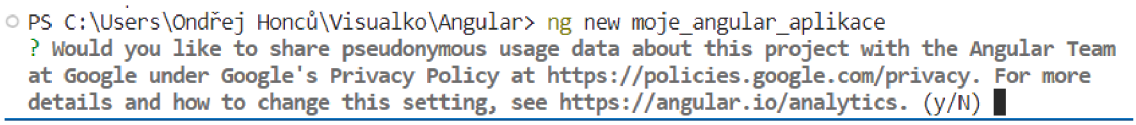
Pro vytvoření webové aplikace za pomoci frameworku Angular je zapotřebí si prvně nainstalovat takzvaný Angular CLI (Command Line Interace), což je nástroj pro vytváření, správu a vývoj Angular aplikací. V terminálu spustíme následující příkaz.

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

Po nainstalování Angular CLI můžete vytvořit novou aplikaci Angular. V terminálu použijte tento příkaz.

```
ng new <název složky>  …. Příklad ng new moje_angular_aplikace
```

Tento příkaz vytvoří novou složku s názvem „moje_angular_aplikace“, která obsahuje strukturu a základní soubory pro novou Angular aplikaci.

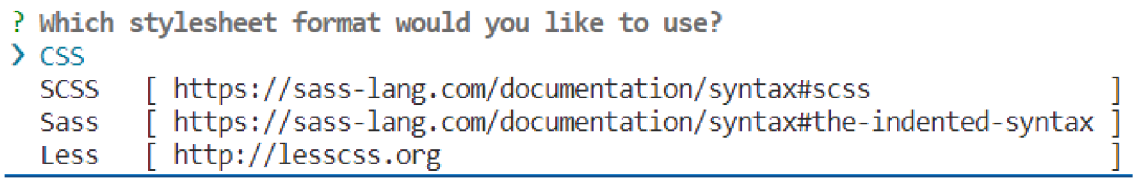


Obrázek 8 Angular, analýza dat
Zdroj: Vlastní zpracování pomocí výstupu z terminálu

Tento terminálový výstup znamená, zda angular team může využívat usage data pro analýzu dat ke zlepšení služeb. Pokud těmto analýzám chceme předejít při tvorbě aplikace, použijeme příkaz s limitací analýz.

```
ng new my_angular_app --skip-analytics
```

Pokud používáme Angular CLI, terminál vyžaduje námi zvolený formát stylů, který chceme použít, máme na výběr ze tří možností: CSS, SCSS, Sass a Less.



Obrázek 9 Angular, stylesheet formát
Zdroj: Vlastní zpracování pomocí výstupu z terminálu

Zde je krátké vysvětlení každé možnosti.

- CSS (Cascading Style Sheets) je standardní formát stylů, který je již dobře znám a používaný webovými vývojáři. Pokud nechceme používat žádný preprocesor stylů, můžeme si vybrat tuto možnost.
- SCSS (Sassy CSS) je rozšíření syntaxe CSS, jež přidává některé vylepšené funkce jako jsou proměnné, vnořené selektory a mnoho dalšího. Je to velmi populární volba pro tvorbu moderních stylů.
- Sass (Syntactically Awesome Style Sheets) je původní syntaxe Sassu, která se od SCSS liší především v zápisu kódu. Místo složených závorek používá odsazení, což některým vývojářům připadá přirozenější.
- Less (Leaner Style Sheets) je dynamický preprocesor kaskádových stylů (CSS), který rozšiřuje syntaxi CSS o funkce jako proměnné, operátory a vnořená pravidla.
  [48]

Dále terminál, přesněji Angular CLI, zjišťuje, zda chcete povolit Server-Side Rendering (SSR) a Static Site Generation (SSG/Prerendering), ptá se na to, zda chcete vytvářet aplikaci s těmito funkcemi.

- Server-Side Rendering (SSR): Je to technika, která umožňuje vykreslování webové stránky na serveru místo v prohlížeči klienta. Tím se zlepšuje výkon a SEO aplikace, ale také se zvyšuje složitost vývoje.
- Static Site Generation (SSG/Prerendering): Jedná se o techniku, jenž generuje statické HTML stránky během sestavení aplikace, což vede ke zvýšení rychlosti načítání a vylepšení výkonu.

Po vytvoření aplikace je nutno přejít do nově vytvořeného adresáře aplikace.

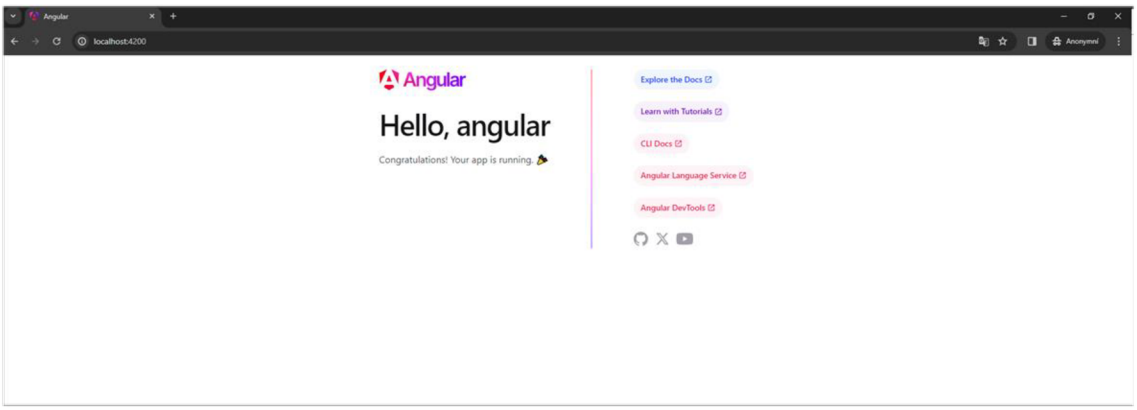
```
cd moje_angular_aplikace
```

Spuštění aplikace

Pro spuštění vývojového serveru pro zobrazení Angular aplikace, využijeme příkaz.

```
ng serve
```

Tento příkaz spustí vývojový server na adrese http://localhost:4200/. Po úspěšném spuštění serveru můžeme v internetovém prohlížeči otevřít tuto adresu a zobrazit aplikaci.



Obrázek 10 Ukázka frameworku Angular 1/2
Zdroj: Vlastní zpracování

Pokud chceme aplikaci upravovat, zasahujeme pouze do souborů ve složce „src“. Změny kódu se automaticky promítají do funkční a běžící aplikace díky automatické obnově vývojového serveru.

Jednoduchá ukázka aplikace

Adresářová struktura projektu

```
appka/
├── src/
│   ├── app/
│   │   ├── app.component.css
│   │   ├── app.component.html
│   │   ├── app.component.spec.ts
│   │   ├── app.component.ts
│   │   ├── app.module.ts
│   ├── assets/
│   │   └── (zde mohou být obrázky nebo jiné soubory)
│   │ ...
├── angular.json
├── package-lock.json
├── package.json
├── README.md
├── tsconfig.app.json
├── tsconfig.json
├── tsconfig.spec.json
└── ...
```

Zdroj: Vlastní zpracování

App.component.ts

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  standalone: true // Přidáme příznak standalone: true
})
export class AppComponent {
  buttonClicked() {
    alert('Klikli jste na tlačítko!');
  }
}
```

Ukázka kódu 16 Angular, app.component.ts
Zdroj: Vlastní zpracování

App.component.html

```
<div class="container">
  <nav>
    <ul>
      <li><a routerLink="/">Domů</a></li>
      <li><a routerLink="/about">O nás</a></li>
      <li><a routerLink="/services">Služby</a></li>
      <li><a routerLink="/contact">Kontakt</a></li>
    </ul>
  </nav>
  <h1 class="title">Vítejte na stránce!</h1>
  <p>Toto je ukázková stránka vytvořená pomocí Angular.</p>
  <button (click)="buttonClicked()" class="button">Klikněte
sem</button>
</div>
```

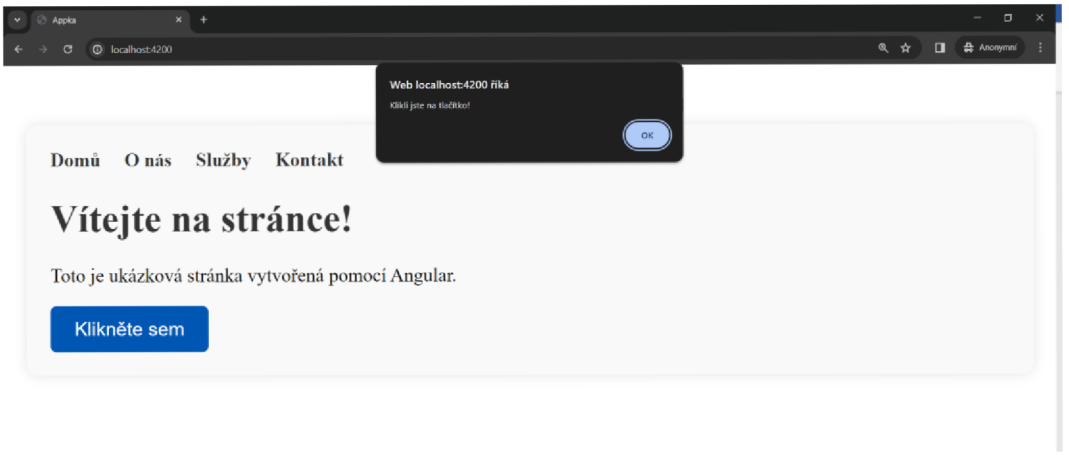
Ukázka kódu 17 Angular, app.component.html
Zdroj: Vlastní zpracování

App.component.css

```
body {
font-family: Arial, sans-serif;
margin: 0;
padding: 0;
}
.container {
max-width: 800px;
margin: 50px auto;
padding: 20px;
background-color: #f9f9f9;
border-radius: 10px;
box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
}
nav ul {
list-style-type: none;
padding: 0;
margin: 0;
}
nav ul li {
display: inline;
margin-right: 20px;
}
nav ul li a {
text-decoration: none;
color: #333;
font-weight: bold;
}
nav ul li a:hover {
color: #007bff;
}
.title {
font-size: 32px;
color: #333;
margin-bottom: 20px;
}
.button {
background-color: #007bff;
color: #fff;
border: none;
padding: 10px 20px;
font-size: 16px;
cursor: pointer;
border-radius: 5px;
}
```

Ukázka kódu 18 Angular, app.component.css
Zdroj: Vlastní zpracování

Vývojový server na adrese http://localhost:4200/.



Obrázek 11 Ukázka frameworku Angular 2/2
Zdroj: Vlastní zpracování

Zdrojové kódy k ukázce Angular aplikace v příloze A.

4.2 React aplikace

Adresářová struktura projektu

```
React/
├── node_modules/
├── public/
│   ├── index.html
├── src/
│   ├── App.css
│   ├── App.js
│   ├── index.css
│   └── index.js
├── server.js
├── package-lock.json
└── package.json
```

Zdroj: Vlastní zpracování

Pro funkčnost aplikace je zapotřebí nainstalovat potřebné komponenty a závislosti. Instalace Reactu v terminálu proběhla pomocí příkazu *npm install react*, následně byl nainstalován i React DOM příkazem *npm install react-dom.* Příkaz *npm install* nainstaluje všechny závislosti projektu uvedené v souboru package.json.

Závěrem, po práci s nástroji a skripty pro vývoj React aplikací, je nutno nainstalovat react-scripts příkazem *npm install react-scripts –save.*

Public – index.html

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';


ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Ukázka kódu 19 React, index.html
Zdroj: Vlastní zpracování

SRC – App.js

```
import React from 'react';
import './App.css';
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <h1>Vítejte v React aplikaci!</h1>
        <p>Toto je jednoduchá aplikace napsaná za použití frameworku React.</p>
      </header>
    </div>
  );
}
export default App;
```

Ukázka kódu 20 React, app.js
Zdroj: Vlastní zpracování

Src-APP.css

```
.App {
    text-align: center;
  }
  .App-header {
    background-color: #282c34;
    min-height: 100vh;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    font-size: calc(10px + 2vmin);
    color: white;
  }
  .App-logo {
    height: 40vmin;
    pointer-events: none;
  }
  @media (prefers-reduced-motion: no-preference) {
    .App-logo {
      animation: App-logo-spin infinite 20s linear;
    }
  }
  @keyframes App-logo-spin {
    from {
      transform: rotate(0deg);
    }
    to {
       transform: rotate(360deg);
     }
  }
```

Ukázka kódu 21 React, app.css
Zdroj: Vlastní zpracování

Src-index.css

```
body {
    margin: 0;
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
      'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
      sans-serif;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  code {
    font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
      monospace;
  }
```

Ukázka kódu 22 React, index.css
Zdroj: Vlastní zpracování

Src-index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

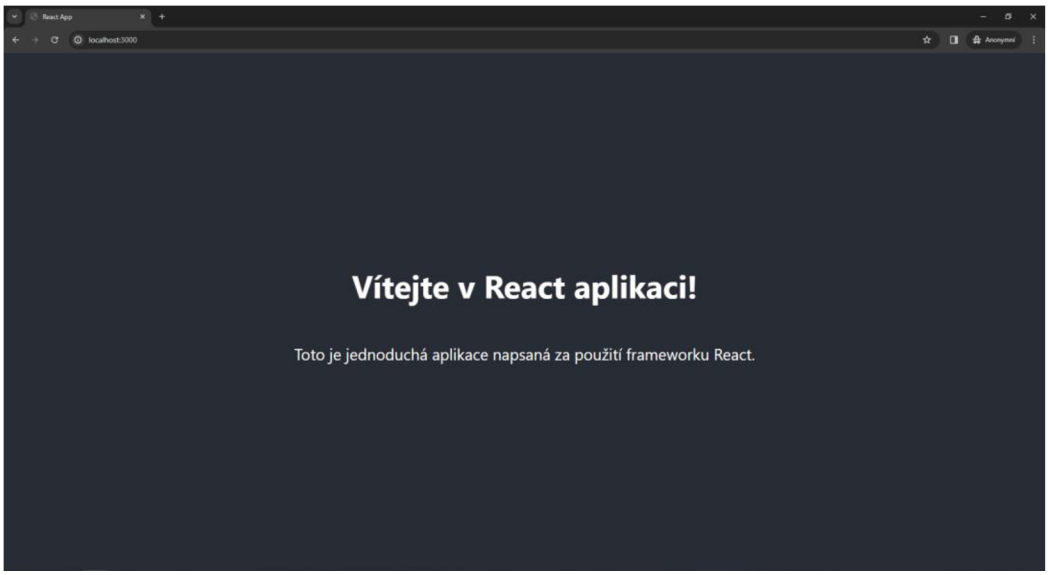
Ukázka kódu 23 React, index.js
Zdroj: Vlastní zpracování

Pro zapnutí aplikace je zapotřebí v terminálu spustit aplikaci pomocí *npm start.*

```
Compiled successfully!

You can now view undefined in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.149:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Obrázek 12 Výstup z terminálu React
Zdroj: Vlastní zpracování

Webový prohlížeč htpp://localhost:3000



Obrázek 13 Ukázka frameworku React
Zdroj: Vlastní zpracování

Zdrojové kódy k ukázce React aplikace v příloze B.

4.3 Vue aplikace

Prvním krokem pro vytvoření aplikace ve Vue.js je instalace Node.js, což lze provést z oficiální stránky Node.js. Následně je potřeba nainstalovat Vue CLI, který je oficiálním nástrojem pro vývoj Vue.js aplikací. Instalaci Vue CLI provedeme pomocí npm (Node Package Manager) příkazu v terminálu.

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

Po instalaci Vue CLI můžeme vytvořit nový projekt pomocí následujícího příkazu.

```
vue create nazev_projektu
```

Během vytváření projektu jsme vyzváni k výběru způsobu tvorby projektu. Defaultní možnost, která obsahuje základní soubory a konfigurace pro Vue.js aplikaci, může být zvolena, nebo můžeme zvolit manuální možnost a projekt nastavit podle vlastních potřeb.

Po vytvoření projektu můžeme začít pracovat na vývoji aplikace. Složka s projektem je otevřena ve vybraném textovém editoru a jsou upravovány soubory ve složce *src*.

Základní struktura Vue.js aplikace obsahuje soubory jako App.vue, který je hlavním komponentem aplikace, a main.js, jenž inicializuje Vue.js a renderuje aplikaci do HTML stránky.

Přednosti Vue.js

Vue.js je známý svou jednoduchostí použití a snadným pochopením. Jeho syntax je čistá a přehledná, což usnadňuje vývoj webových aplikací. Díky tomu je ideální pro začínající vývojáře i pro rozsáhlejší projekty.

```
<div id="app">
  {{ message }}
</div>
<script>
  var app = new Vue({
    el: '#app',
    data: {
      message: 'Hello Vue!'
    }
  });
</script>
```

Ukázka kódu 24 Vue, Hello Vue
Zdroj: Vlastní zpracování

Vue.js poskytuje velkou flexibilitu při tvorbě webových aplikací. Může být integrován postupně do existujícího projektu nebo použit jako plnohodnotný framework pro vývoj SPA. Díky modulární architektuře lze Vue.js snadno rozšiřovat o další funkcionality.

```
<template>
  <div>
    <h1>{{ greeting }}</h1>
    <button @click="changeGreeting">Change Greeting</button>
  </div>
</template>

<script>
export default {
  data() {
    return {
      greeting: 'Hello Vue!'
    };
  },
  methods: {
    changeGreeting() {
      this.greeting = 'Bonjour Vue!';
    }
  }
}
</script>
```

Ukázka kódu 25 Vue, funkcionality
Zdroj: Vlastní zpracování

Ukázka Aplikace za pomoci Vue.js

Pokud proběhlo správné vytvoření Vue.js aplikace, stačí pouze přepisovat soubor App.vue. Pro ukázku aplikace bylo vybráno téma „to do listu“. Aplikace se skládá z App.cue <template>, App.vue <script>, App.vue <style scoped>

App.vue <template>

```
<template>
 <div id="app" class="app-container">
  <div class="todo-container">
   <h1 class="app-title">Seznam úkolů!</h1>
   <div class="todo-form">
    <input v-model="newTask" @keyup.enter="addTask" placeholder="Přidat nový úkol"
class="task-input">
    <button @click="addTask" class="add-button">Přidat úkol</button>
   </div>
   <ul class="todo-list">
    <li v-for="(task, index) in tasks" :key="index" class="todo-item">
     <input type="checkbox" v-model="task.completed" class="task-checkbox">
     <span :class="{ completed: task.completed }" class="task-text">{{ task.text }}</span>
     <button @click="deleteTask(index)" class="delete-button">X</button>
    </li>
   </ul>
  </div>
 </div>
</template>
```

Ukázka kódu 26 App.vue <template>
Zdroj: Vlastní zpracování

App.vue <script>

```
<script>
export default {
 name: 'App',
 data() {
  return {
   newTask: '',
   tasks: []
  };
 },
 methods: {
  addTask() {
   if (this.newTask.trim() !== '') {
    this.tasks.push({ text: this.newTask, completed: false });
    this.newTask = '';
   }
  },
  deleteTask(index) {
   this.tasks.splice(index, 1);
  }
 }
}
</script>
```

Ukázka kódu 27 App.vue <script>
Zdroj: Vlastní zpracování

App.vue <style scoped> část 1/2

```
<style scoped>
.app-container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background-color: #f5f5f5;
}
.todo-container {
  width: 400px;
  padding: 20px;
  border-radius: 10px;
  background-color: white;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.app-title {
  margin-bottom: 20px;
  text-align: center;
  color: #333;
}

.todo-form {
  display: flex;
  margin-bottom: 20px;
}

.task-input {
  flex: 1;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px 0 0 5px;
}
```

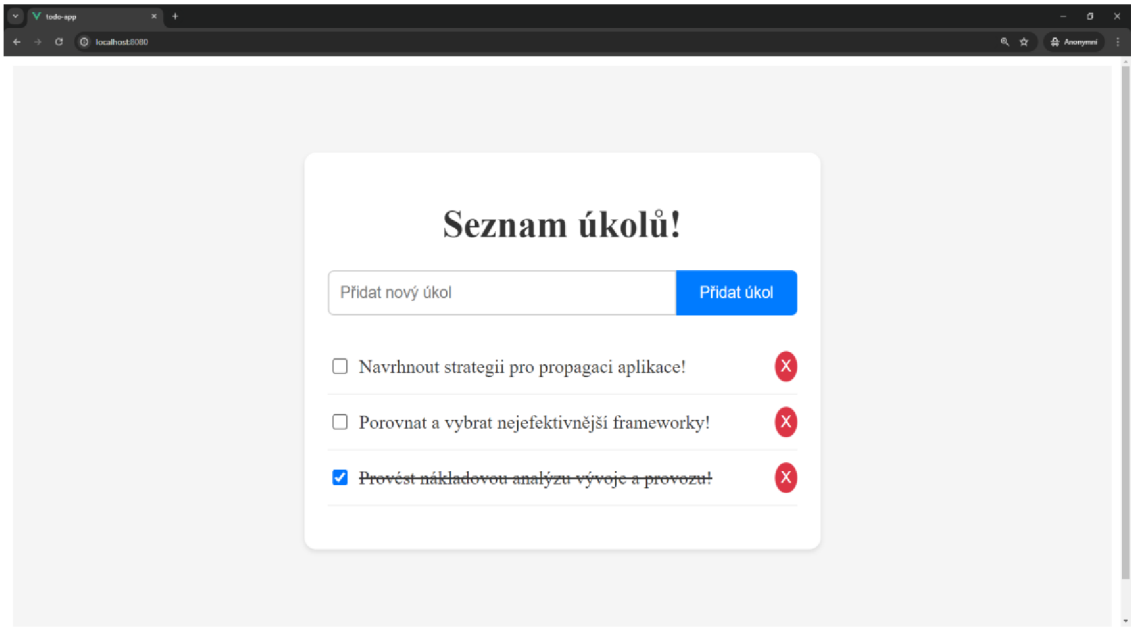
Ukázka kódu 28 App.vue <style scoped> část 1/2
Zdroj: Vlastní zpracování

App.vue <style scoped> část 2/2

```
.add-button {
  padding: 10px 20px;
  background-color: #007bff;
  border: none;
  border-radius: 0 5px 5px 0;
  color: white;
  cursor: pointer;
}
.todo-list {
  list-style: none;
  padding: 0;
}
.todo-item {
  display: flex;
  align-items: center;
  border-bottom: 1px solid #eee;
  padding: 10px 0;
}
.task-checkbox {
  margin-right: 10px;
}
.task-text {
  flex: 1;
  color: #333;
}
.completed {
  text-decoration: line-through;
}
.delete-button {
  padding: 5px;
  background-color: #dc3545;
  border: none;
  border-radius: 50%;
  color: white;
  cursor: pointer;
}
</style>
```

Ukázka kódu 29 App.vue <style scoped> část 2/2
Zdroj: Vlastní zpracování

Pro spuštění vytvořené aplikace použijeme příkaz *npm run serve* v teminálu. Tento příkaz spouští vývojový server na localhostu a zobrazuje adresu http://localhost:8080/ , na které bude aplikace dostupná v prohlížeči.



Obrázek 14 Ukázka frameworku Vue
Zdroj: Vlastní zpracování

Zdrojové kódy k ukázce Vue aplikace v příloze C.

4.4 Svelte aplikace

Pro vytvoření aplikace ve Svetle.js je třeba Node.js. Následně je potřeba nainstalovat Svelte pomocí *npx degit sveltejs/template svelte-app*. Tento příkaz vytvoří nový projekt Svelte pomocí šablony poskytnuté týmem Svelte. Degit je nástroj pro kopírování repozitáře, který je součástí balíčku npx. Tímto příkazem získáme základní strukturu projektu Svelte s potřebou stažení Gitu. Použijeme *cd svelte-app.* Příkaz cd (change directory) změní pracovní adresář do složky svelte-app, kterou jsme vytvořili pomocí předchozího příkazu. Příkaz *npm install* nainstaluje všechny závislosti (knihovny a balíčky), které jsou uvedené v souboru package.json v projektu. Tyto závislosti zahrnují Svelte a další potřebné balíčky pro vývoj aplikace.

Adresářová struktura aplikace Svelte.js

```
svelte-app/
├── node_modules/
├── public/              # Veřejné soubory (HTML, CSS, obrázky)
│   ├── index.html       # Vstupní HTML soubor
│   └── favicon.ico      # Ikona webové stránky
├── scripts/             # Nastavení typescriptu
├── src/                 # Zdrojové soubory aplikace
│   ├── App.svelte       # Hlavní komponenta aplikace
│   ├── Task.svelte      # Komponenta pro zobrazení jednotlivých úkolů
│   └── ...              # Další zdrojové soubory
├── .gitignore
├── package-lock.json    # Závislosti s přesnými verzemi (generovaný npm)
├── package.json         # Konfigurační soubor pro Node.js projekt
└── README.md            # Soubor s popisem projektu
```

Zdroj: Vlastní zpracování

App.svelte

Kód pro App.svelte `<script>`

```
<script>
    // Importy funkcí z balíčku Svelte a komponenty Task
    import { onMount } from 'svelte';
    import { writable } from 'svelte/store';
    import Task from './Task.svelte';

    // Vytvoření reaktivní proměnné pro seznam úkolů a inicializace
prázdným polem
    let tasks = writable([]);
    // Proměnná pro ukládání názvu nového úkolu z formuláře
    let newTaskTitle = '';
  // Funkce pro přidání nového úkolu do seznamu úkolů
    function addTask() {
        // Kontrola, zda je název nového úkolu neprázdný
      if (newTaskTitle.trim()) {
        // Aktualizace seznamu úkolů - přidání nového úkolu
        tasks.update(arr => [...arr, { title: newTaskTitle, completed:
false }]);
        // Vyprázdnění pole pro nový název úkolu
        newTaskTitle = '';
      }
    }
  // Funkce pro odstranění úkolu ze seznamu na základě indexu
    function deleteTask(index) {
        // Aktualizace seznamu úkolů - odstranění úkolu s
odpovídajícím indexem
      tasks.update(arr => arr.filter((_, i) => i !== index));
    }
  // Funkce pro označení úkolu jako dokončeného nebo nedokončeného na
základě indexu
    function completeTask(index) {
        // Aktualizace stavu úkolu - změna hodnoty
      tasks.update(arr => {
        arr[index].completed = !arr[index].completed;
        return arr;
      });
    }
  </script>
```

Ukázka kódu 30 App.svelte <script>
Zdroj: Vlastní zpracování

App.svelte

Kód pro App.svelte `<style>`.

```
<style>
    /* Styly pro hlavní kontejner aplikace */
    .app-container {
      max-width: 600px;
      margin: 0 auto;
      padding: 20px;
      border: 1px solid #ddd;
      border-radius: 10px;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    }
    /* Styly pro nadpis */
    h1 {
      text-align: center;
      margin-bottom: 20px;
      font-size: 24px;
      color: #333;
    }
  /* Styly pro vstupní pole */
    input[type="text"] {
      width: calc(100% - 80px);
      padding: 10px;
      font-size: 16px;
      border: 1px solid #ddd;
      border-radius: 5px;
      margin-bottom: 10px;
    }
    button {
      padding: 10px 20px;
      font-size: 16px;
      background-color: #007bff;
      color: #fff;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }
    button:hover {
      background-color: #0056b3;
    }
  </style>
```

Ukázka kódu 31 App.svelte <style>
Zdroj: Vlastní zpracování

App.svelte

Kód pro App.svelte `<div>`.

```
<div class="app-container">
    <h1>Seznam úkolů</h1>
    <!-- Formulář pro přidání nového úkolu -->
    <div style="display: flex; margin-bottom: 20px;">
      <input type="text" bind:value={newTaskTitle} placeholder="Přidat
nový úkol">
      <button on:click={addTask}>Přidat úkol</button>
    </div>
  <!-- Seznam úkolů s použitím komponenty Task -->
    {#each $tasks as task, index}
      <Task {task} onDelete={() => deleteTask(index)} onComplete={()
=> completeTask(index)} />
    {/each}
  </div>
```

**Ukázka kódu 32 App.svelte **
Zdroj: Vlastní zpracování

Task.svelte

Kód pro Task.svelte.

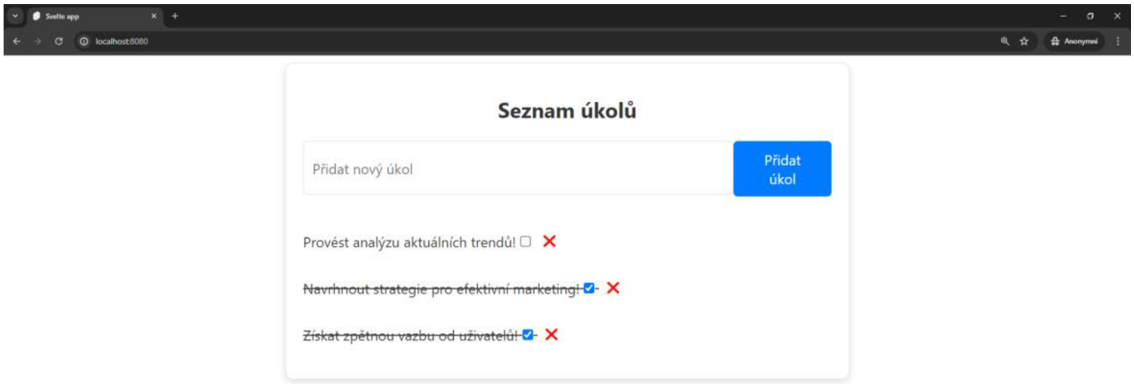
```
<script>
    // Exportované proměnné pro předání dat a funkcí z rodičovské
komponenty
    export let task;        // Informace o konkrétním úkolu
    export let onDelete;    // Funkce pro smazání úkolu
    export let onComplete;  // Funkce pro označení úkolu jako dokončený/nedokončený
</script>
<div class="{task.completed ? 'completed' : ''}">
    <span>{task.title}</span>
    <input type="checkbox" checked="{task.completed}"
on:change={onComplete}>
    <button class="delete-btn" on:click={onDelete}>✕</button>
</div>
<style> /* Různé styly */
    .completed {
        text-decoration: line-through;
    }
    .delete-btn {
        background: none;
        border: none;
        color: red;
        font-size: 20px;
        cursor: pointer;
    }
</style>
```

Ukázka kódu 33 Task.svelte
Zdroj: Vlastní zpracování

Nastavení projektu a serveru s jejím spuštěním

Než začnete, ujistěte se, že máte projekt připravený a všechny potřebné závislosti nainstalované. To zahrnuje závislosti na serverovém softwaru, pokud je potřebujete, stejně jako veškerý kód a soubory vaší aplikace**.** Poté, co máte projekt vytvořený a nainstalované závislosti, můžete spustit vývojový server pomocí příkazu *npm run dev.* Po úspěšném spuštění vývojového serveru můžete přistupovat k vaší Svelte aplikaci pomocí webového prohlížeče na adrese http://localhost:8080 .

Ukázka aplikace na localhostu.



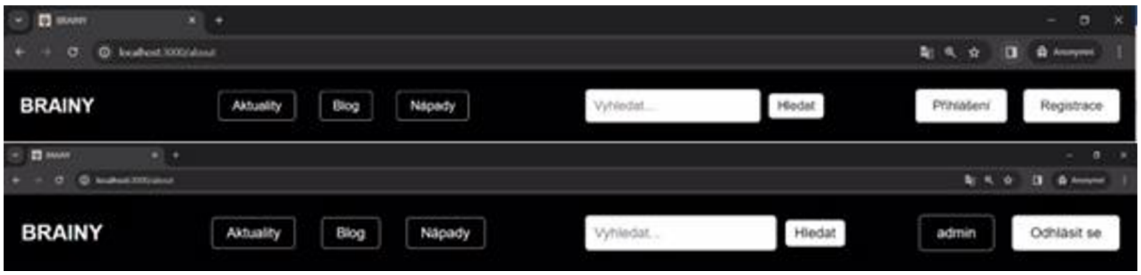
Obrázek 15 Ukázka frameworku Svelte
Zdroj: Vlastní zpracování

Zdrojové kódy k ukázce Svelte aplikace v příloze D.

5 Rozšíření aplikace

Po zpracování informací z předchozích kapitol a tvorby v jednotlivých frameworcích byla rozšířená frontendová aplikace vyvíjena jako Single Page Aplikace (SPA) v JavaScriptovém frameworku React.js. Aplikace byla vytvořena pro ukázku tvorby webového designu pro aplikace.

5.1 Navigační lišta



Obrázek 16 Navigační lišta React
Zdroj: Vlastní zpracování

Navigační panel aplikace by měl být navržen tak, aby poskytoval uživatelům intuitivní a přehledný způsob navigace na všech stránkách. V ukázkové aplikaci existují dvě varianty navigačního panelu, jedna pro nepřihlášeného uživatele a druhá pro přihlášeného uživatele. Tento designový přístup je zvolen z důvodu optimalizace uživatelského zážitku a zvýšení přehlednosti.

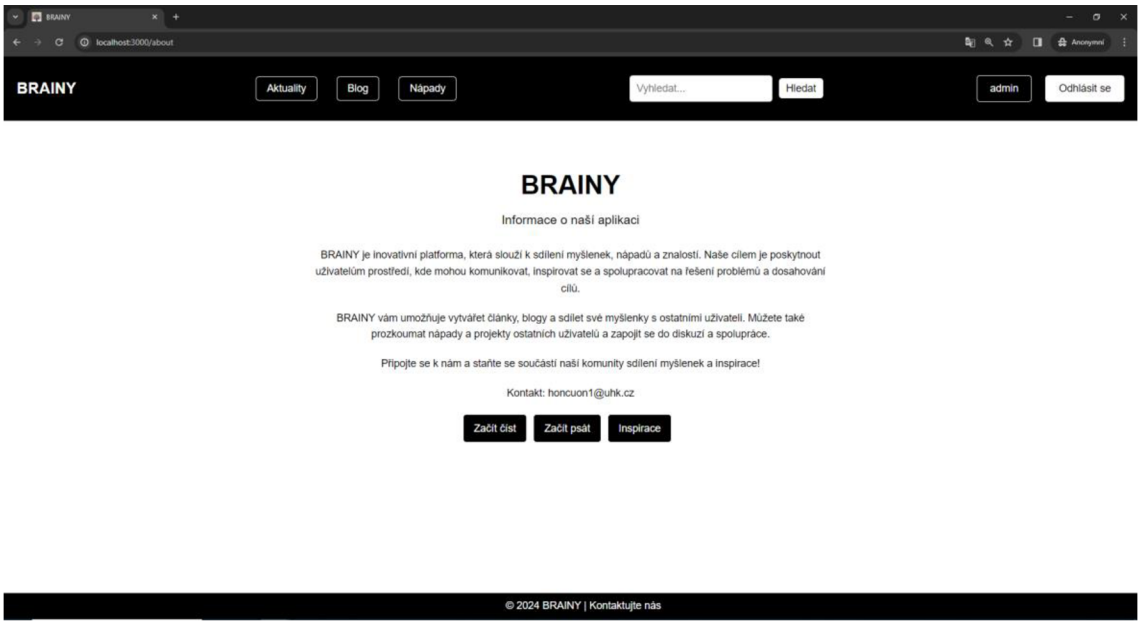
Pro nepřihlášené uživatele je na pravé straně navigačního panelu umístěna možnost přihlášení či registrace, což umožňuje snadný přístup k účtu a jeho správě. Tímto způsobem je uživateli umožněno rychleji přistoupit k funkcím, které souvisejí s jeho účtem, a tím zlepšit uživatelský zážitek. Na druhou stranu, přihlášený uživatel vidí své uživatelské jméno a možnost odhlášení. Tento přístup zvyšuje personalizaci uživatelského rozhraní a umožňuje uživateli snadněji identifikovat, zda je přihlášen na stránce.

Oba typy uživatelů by měly mít přístup k hlavním kategoriím aplikace, jako jsou zde Aktuality, Blog a Nápady. Pro lepší orientaci na stránce či v aplikaci je dobré vytvořit vyhledávací lištu, která napomáhá vyhledání dané informace. Tato

konzistentní struktura navigace usnadňuje uživatelům orientaci na stránce a umožňuje jim snadno přecházet mezi různými částmi aplikace.

5.2 Úvodní strana

AboutPage je důležitou součástí pro poskytnutí přehledného a informativního základu uživatelům o tom, co aplikace nabízí a jak ji mohou využít.



Obrázek 17 Úvodní strana React
Zdroj: Vlastní zpracování

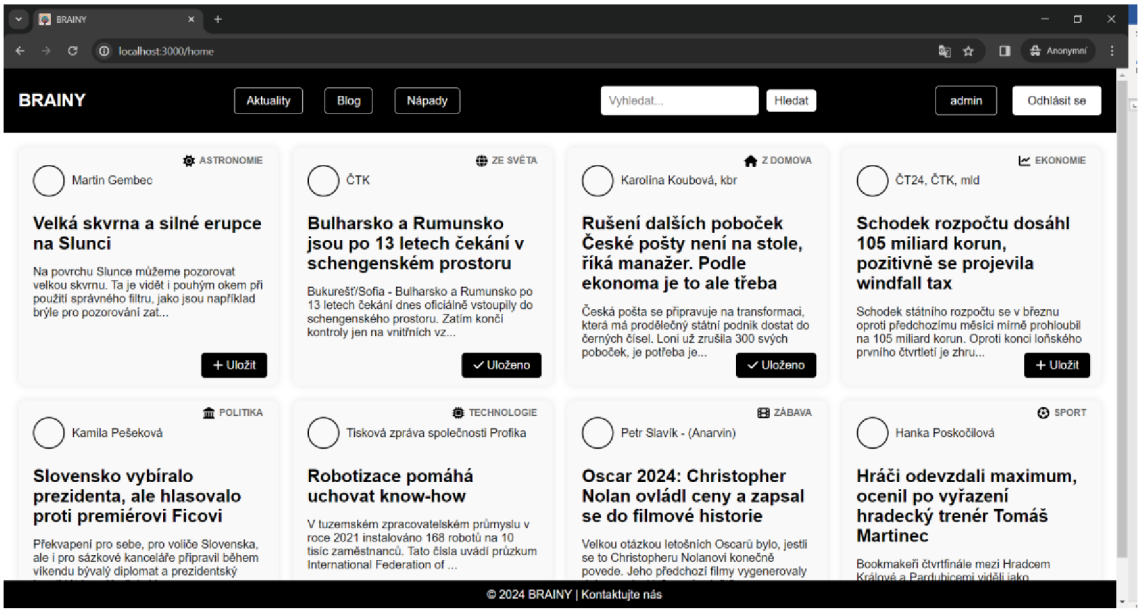
AboutPage by měl být takový stručný přehled o tom, co by měl očekávat v aplikaci. Obsahuje jasný popis cílů a funkcí aplikace, psaný srozumitelným jazykem a prezentovaný strukturovaně, aby byl uživatelům snadno přístupný a pochopitelný. Tímto způsobem se snažíme zajistit, že uživatelé rychle a snadno pochopí, co BRAINY nabízí a jaké jsou jeho hlavní výhody.

Vložení tlačítek na About stránku poskytuje uživatelům jedinečnou příležitost prezentovat svou aplikaci nejen slovy, ale i akcemi. Tato tlačítka slouží jako navigační prvky, které umožňují návštěvníkům rychlý a snadný přístup k hlavním funkcím nebo obsahu aplikace. Zatímco textový popis může poskytnout obecný přehled, tlačítka dodávají interaktivitu a umožňují uživatelům přímé přechody do důležitých částí aplikace.

5.3 Obsah aplikace

Obsah webové aplikace by měl být prezentován tak, aby poskytoval uživatelům snadný přístup k informacím a usnadňoval jim interakci s aplikací. Klíčovým prvkem je strukturovaný layout, který rozděluje obsah do jasně definovaných sekcí nebo bloků. Každá sekce by měla mít svůj vlastní nadpis a vizuálně oddělený prostor, což pomáhá uživatelům snadno se orientovat na stránce.

Aktuality



Obrázek 18 Aktuality React
Zdroj: Vlastní zpracování za použití textu [23][24][25][26][27][28][29][30]

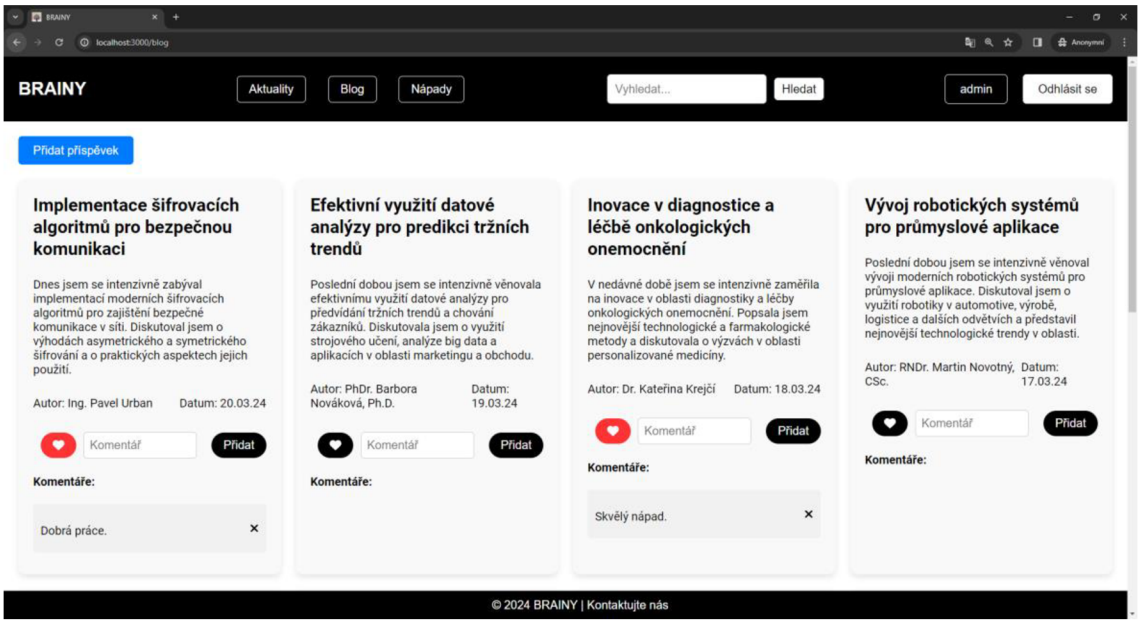
Důležité je také formátování obsahu. Text by měl být členěn do krátkých odstavců, aby se zlepšila čitelnost, a měl by využívat nadpisů, odrážek a číslovaných seznamů k zdůraznění klíčových informací. Tímto způsobem je obsah strukturovaný a lépe organizovaný.

Vizuální prvky, jako jsou grafika, obrázky a ikony, jsou dalším důležitým prvkem obsahu webové aplikace. Pokud jsou použité správně, mohou zlepšit vizuální atraktivitu a přispět k lepšímu porozumění obsahu. Nicméně je důležité používat je s mírou a zohlednit jejich relevanci k obsahu. Kromě toho by měl obsah obsahovat jasný a srozumitelný jazyk. Texty by měly být psány tak, aby byly přístupné

a snadno pochopitelné pro cílovou skupinu uživatelů. To znamená minimalizovat používání složitých termínů a vysvětlovat nejasné pojmy, pokud se nejedná o aplikaci k tomu určenou. Důraz by měl být kladen na komunikaci srozumitelným a přátelským jazykem, který zlepšuje uživatelskou zkušenost a usnadňuje interakci s aplikací.

Blog

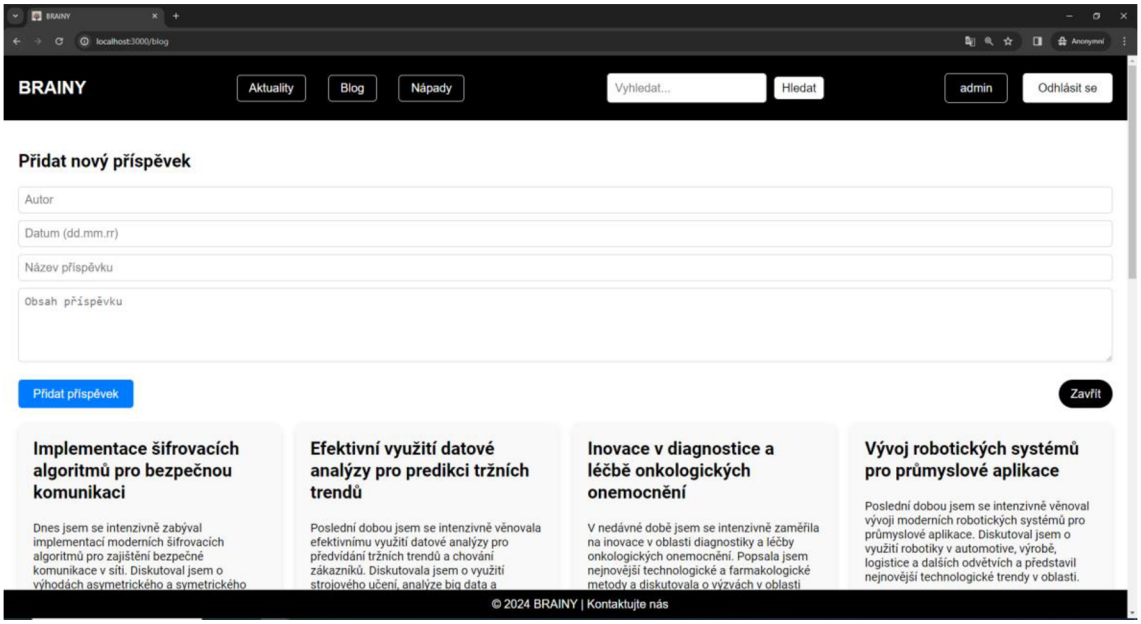
Intuitivní navržení je klíčové pro vytvoření uživatelsky přívětivé aplikace. Účelem je poskytnout uživatelům pohodlný a efektivní způsob pro procházení a sdílení obsahu. Uživatelé by měli mít možnost snadno procházet obsah a rychle nalézt ty, které jsou pro ně relevantní. Intuitivní rozvržení a navigační prvky, jako jsou titulky a ovládací tlačítka, usnadňují uživatelům orientaci v obsahu a zkracují dobu potřebnou k nalezení požadovaných informací. Uživatelé by měli mít možnost interagovat s obsahem prostřednictvím různých akcí, jako je označení článků jako oblíbené, jejich úprava nebo dokonce jejich smazání.



Obrázek 19 Blog React
Zdroj: Vlastní zpracování

Dbejte na jednoduchost procesu přidávání obsahu. Pro uživatele, kteří chtějí přispívat svými články, by měl být proces přidání nového obsahu snadný a přímočarý. Intuitivní formulářové prvky a jasné instrukce usnadňují uživatelům vytváření a publikování svých příspěvků.

Tato interaktivita přispívá k zapojení uživatelů a zlepšuje jejich celkový zážitek z používání aplikace. Vytvoření s důrazem na uživatelskou intuitivnost přispívá k celkové kvalitě aplikace a zvyšuje uživatelskou spokojenost. Uživatelé si váží jednoduchosti a přístupnosti, což má pozitivní dopad na popularitu a uživatelskou základnu aplikace.

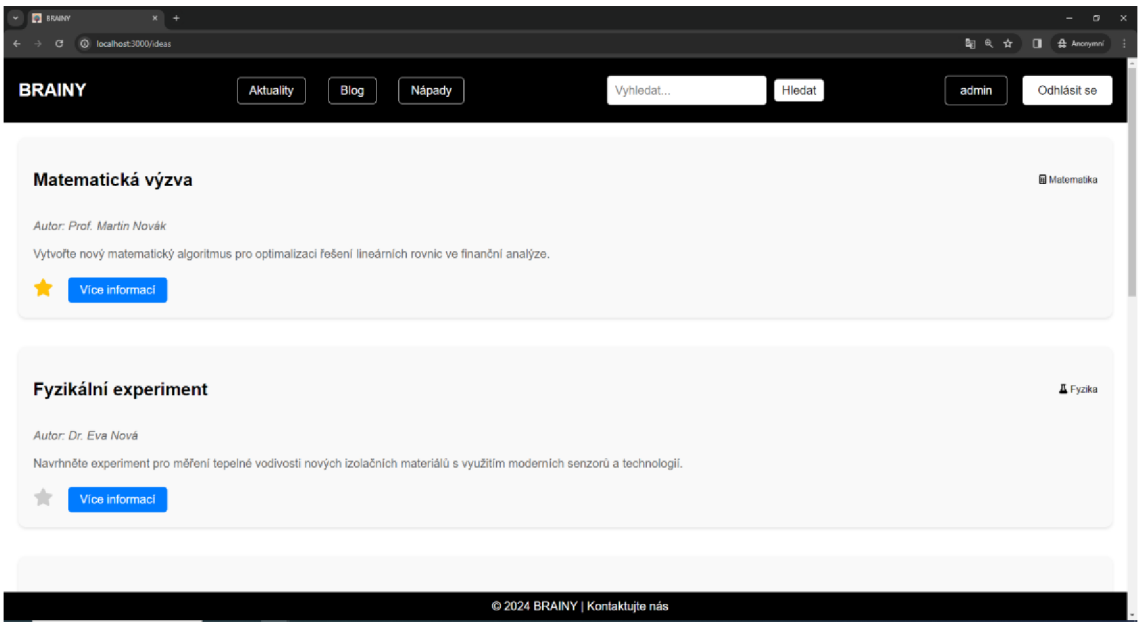


Obrázek 20 Přidání nového příspěvku React
Zdroj: Vlastní zpracování

Nápady

Poslední ukázková stránka je navržena tak, aby byla uživatelům co nejvíce přehledná a jednoduchá. Každý záznam na stránce je připraven tak, aby mohl být snadno identifikován a prozkoumán, přičemž je klíčové, aby byla zachována jasnost a srozumitelnost pro uživatele. Významně se využívají ikony, aby byl posílen profesionální vzhled a rychlá identifikace kategorií, což přispívá k uživatelskému komfortu a pohodlí.

Co se týče kódu, je navržen tak, aby byl co nejčitelnější a snadno upravitelný. Každý nápad je reprezentován jako samostatná komponenta, což usnadňuje úpravy a rozšiřování funkčnosti aplikace. Takto navržený kód zajišťuje, že tvorba a údržba aplikace budou efektivní a bezproblémové.



Obrázek 21 Nápady React
Zdroj: Vlastní zpracování

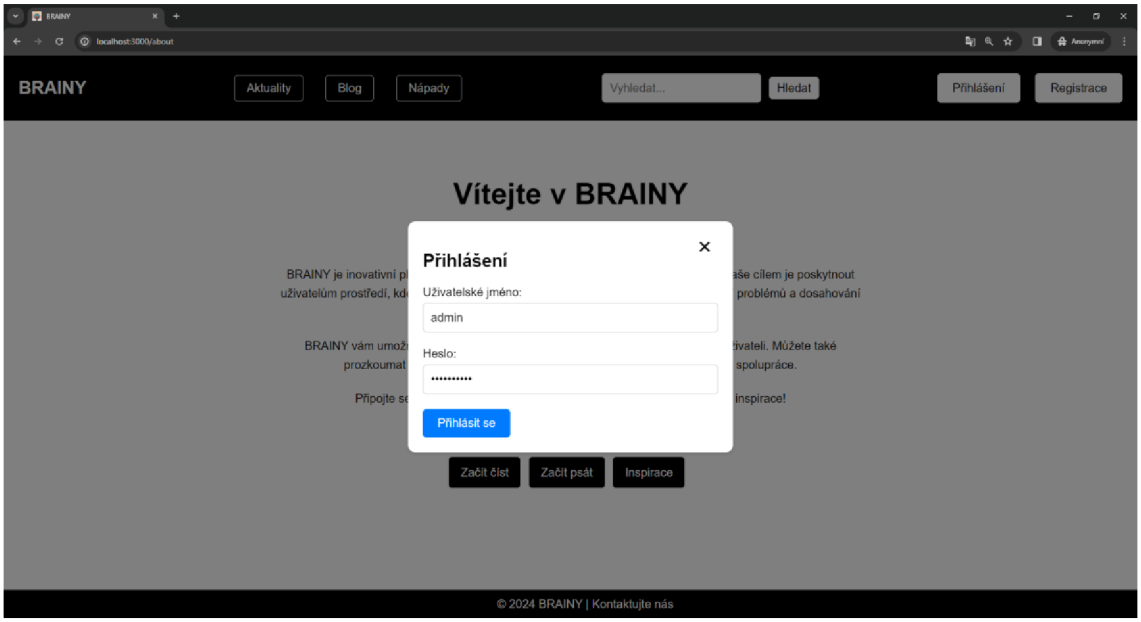
5.4 Modální okno

Modální okno je užitečný prvek uživatelského rozhraní, který se často využívá k zobrazení důležitých informací, interaktivních formulářů nebo potvrzení akcí, jež vyžadují okamžitou reakci od uživatele. Jedná se o dialogové okno, které se objevuje před hlavním obsahem stránky a dočasně zastavuje interakci s ostatními prvky, dokud není vyřešeno. Modální okno obvykle obsahuje výrazné nadpisy, jež jasně informují uživatele o účelu okna, a případně i ilustrační prvky nebo ikony pro lepší vizuální orientaci.

Důležitou vlastností modálních oken je možnost zavření, a to buď pomocí tlačítka „Zavřít“ nebo stisknutím klávesy Escape. Tím se uživateli umožňuje vrátit se k hlavnímu obsahu stránky nebo pokračovat v interakci s ostatními prvky. Modální okna jsou často využívána pro potvrzení akcí, jako je například smazání položky nebo potvrzení odeslání formuláře, přihlášení či registrace. Jsou také užitečná pro

poskytnutí dalších informací nebo podrobností, které by mohly být užitečné pro uživatele při rozhodování. Výhodou modálních oken je, že umožňují koncentraci uživatele na konkrétní úkol nebo informaci, aniž by byl rušen ostatními prvky na stránce. Jsou také flexibilní a snadno přizpůsobitelné různým potřebám a designovým požadavkům aplikace.

Využití

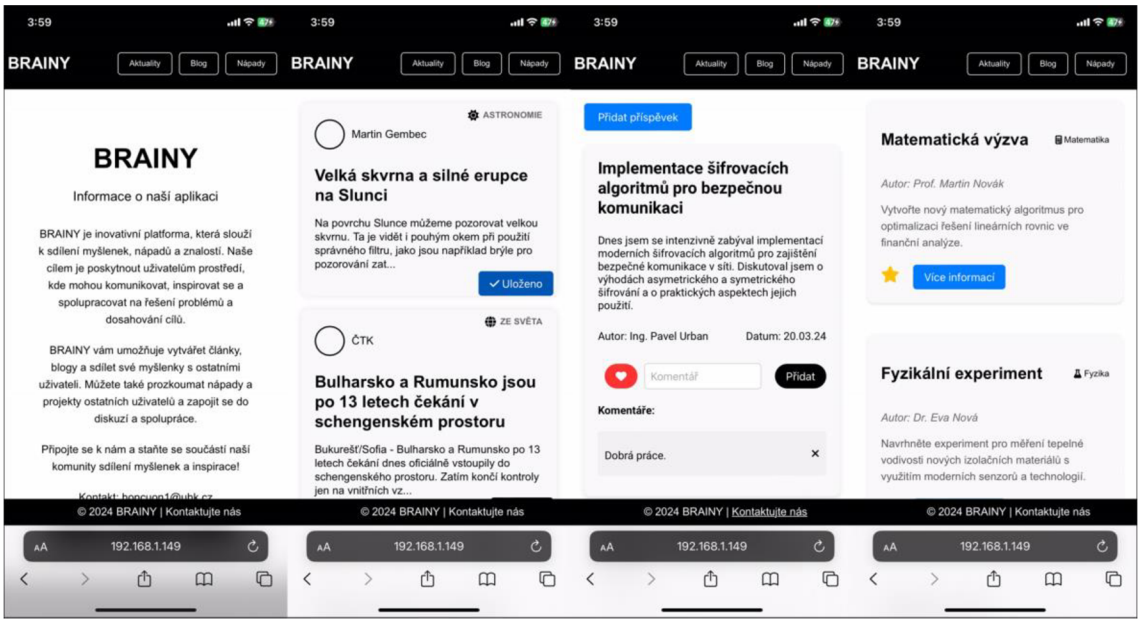


Obrázek 22 Modální okno React
Zdroj: Vlastní zpracování

Modální okno bylo využito při registraci a přihlášení uživatele z několika důvodů. Prvním důvodem je poskytnutí jasně vymezeného prostoru pro zadání přihlašovacích údajů uživatelem. Tímto způsobem je zamezeno rozptýlení pozornosti uživatele a umožněno se snáze zaměřit na vyplnění potřebných informací. Zdůrazňuje důležitost a naléhavost, což může posílit pocit důležitosti procesu registrace a přihlašování. Uživatelé jsou tak motivováni k dokončení procesu registrace nebo přihlášení. Dalším důležitým faktorem je také jednoduchost navigace. Modální okno se objevuje přímo na středu obrazovky, což zajišťuje snadný a rychlý přístup bez nutnosti přecházení na jinou stránku.

5.5 Responzivní design

Použití responzivního designu je klíčové pro vytvoření webové stránky, která se přizpůsobí různým zařízením, velikostem obrazovek a potřebám uživatele. Responzivní design zajišťuje optimální zobrazení a uživatelský komfort na jakémkoli zařízení, což vede k lepší uživatelské zkušenosti a zvýšenému uživatelskému působení a návštěvnosti. Ideální je optimalizovat rozložení obrazovky pro různá zařízení, použít menší zobrazení pro mobilní uživatele, střední pro uživatele tabletů a standardní pro uživatele počítačů. Responzivního designu lze dosáhnout pomocí media queries @media, které umožňuje definovat různé styly na základě šířky obrazovky. Například v kódu @media screen and (max-width: 730px) jsou definovány styly pro zobrazení navigace ve sloupci a skrytí vyhledávacího pole a uživatelských akcí na zařízeních s menší šířkou obrazovky. Na větších obrazovkách dochází k navrácení uživatelských akcí. Tímto způsobem se zajistí, že navigace a ostatní prvky budou pohodlně a efektivně viditelné na různých zařízeních. Dále responzivní design umožňuje webovým stránkám dosáhnout lepšího vyhledávání na vyhledávacích zařízeních a lepší pozice ve výsledcích vyhledávání. To vede ke zvýšené návštěvnosti a viditelnosti stránky.



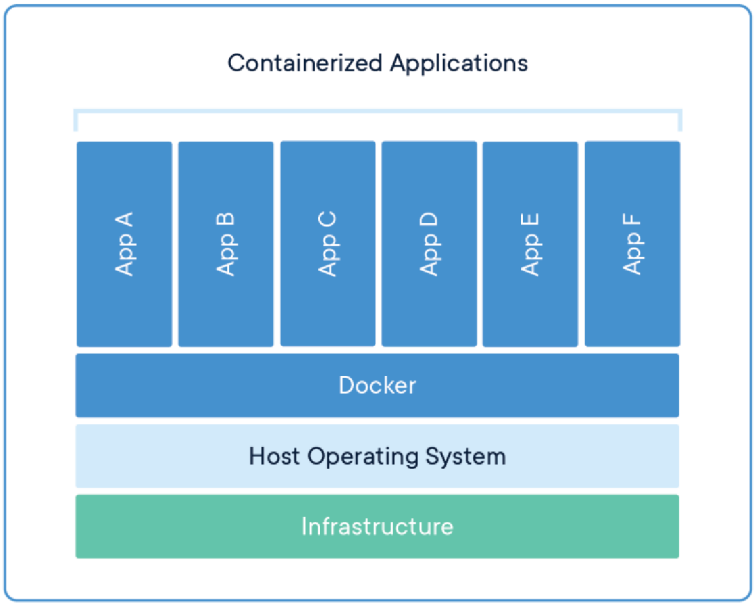
Obrázek 23 Responzivní design React
Zdroj: Vlastní zpracování

6 Zpřístupnění aplikace

Pro zpřístupnění ukázkové aplikace bylo využito platformy docker.

6.1 Docker

Docker je platforma pro kontejnerizaci aplikací, což je technologie umožňující vytváření, spouštění a řízení kontejnerů. Kontejnerizace je proces, který umožňuje zapouzdřit aplikaci spolu s jejími závislostmi do izolovaného prostředí nazývaného kontejner. Tím se vytváří jednotný a konzistentní prostředek pro distribuci aplikací napříč různými prostředími, od vývojářských stanic až po produkční servery. Docker umožňuje vytvořit tzv. Docker obrazy (images), které obsahují veškeré potřebné komponenty pro spuštění aplikace, včetně operačního systému, knihoven, frameworků a dalších závislostí. Tyto obrazy jsou snadno reprodukovatelné a přenosné, což znamená, že aplikace nasazené pomocí Dockeru budou pracovat konzistentně napříč různými prostředími, bez ohledu na to, zda se jedná o vývojářský notebook, vzdálený server nebo cloudovou platformu. Jednou z hlavních výhod Dockeru je izolace aplikace v kontejneru. To zajišťuje, že aplikace nemá přímý vliv na hostitelský operační systém a že jsou kontejnery nezávislé a izolované, což minimalizuje riziko konfliktů a problémů spojených s instalací a spouštěním aplikací. [45]



Obrázek 24 Docker
Zdroj: docker.com [45]

Dockerfile

```
FROM node:20.11-alpine as builder  # Použití obrazu Node.js jako
základu pro první fázi buildu
WORKDIR /app  # Nastavení pracovního adresáře v kontejneru na /app

# Kopírování package.json, package-lock.json a ostatní soubory v
adresáři do kontejneru
COPY ["package.json", "package-lock.json", "./"]

# Instalace závislostí (vynechání vývojových závislostí)
RUN npm install --omit=dev

# Kopírování zdrojových souborů do kontejneru
COPY . .

# Spuštění build procesu
RUN npm run build

FROM nginx:alpine  # Použití obrazu NGINX jako základu pro druhou fázi
COPY --from=builder /app/build ./usr/share/nginx/html  # Kopírování
výstupního buildu z první fáze do adresáře s HTML obsahem NGINXu
EXPOSE 80  # Exponování portu 80
CMD ["nginx", "-g", "daemon off;"]  # Spuštění NGINXu
```

Ukázka kódu 34 Dockerfile
Zdroj: Vlastní zpracování

Docker compose

```
version: '3'  # Verze formátu souboru docker-compose.yml
services:  # Definice služeb (kontejnerů)
  frontend:  # Jméno služby
    container_name: "brainy"  # Název kontejneru
    build:  # Definice postupu sestavení obrazu kontejneru
      context: .  # Aktuální adresář jako kontext pro build
      dockerfile: Dockerfile  # Jméno souboru Dockerfile pro sestavení
obrazu
    ports:  # Mapování portů mezi hostitelským počítačem a kontejnerem
      - "3000:80"  # Port 3000 na hostiteli mapován na port 80 v
kontejneru
    volumes:  # Připojení externího volume (adresáře) do kontejneru
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf  #
Připojení konfiguračního souboru NGINXu
```

Ukázka kódu 35 Docker compose
Zdroj: Vlastní zpracování

Návod na spuštění aplikace

1. Nainstalovat docker engine (ke stažení na adrese: docker.com)
2. Stáhnutí potřebných souborů z adresy: https://github.com/Zeddrake/Brainy,
3. Spustit terminál ve složce obsahující zdrojové soubory a soubor docker-compose.yml.
4. Spuštění příkazu *docker compose up*
5. Webová aplikace nyní běží na adrese http://localhost:3000

7 Shrnutí výsledků

Práce byla zaměřena na analýzu moderních frameworků určených pro základní tvorbu a design webových aplikací. Hlavním cílem bylo nejenom představit tyto frameworky, ale také ukázat, jakým způsobem se s nimi pracuje v praxi. Prostřednictvím praktických ukázek kódu byly detailněji objasněny základní principy, struktura a rozdíly mezi jednotlivými frameworky. Nebyly poskytnuty pouze teoretické znalosti, ale také praktické dovednosti, které jsou nezbytné pro úspěšné vytvoření webových aplikací. Dalším důležitým aspektem bylo provedení rozšíření webové aplikace ve vybraném frameworku na konkrétním modelovém příkladu. Tím bylo demonstrováno, jak se jednotlivé frameworky liší v použití a jak lze jejich funkce efektivně využít v praxi.

Cílem také bylo vytvoření uceleného návodu, který by čtenářům umožnil snadno a rychle seznámit se s tvorbou webových aplikací v různých frameworcích. Věřím, že tato práce poskytuje užitečné informace a inspiraci pro všechny, kteří se zajímají o vývoj moderních webových aplikací. Při tvorbě aplikace bylo dbáno na to, aby byla přístupná a pohodlná pro uživatele na různých zařízeních, včetně počítačů, tabletů a mobilních telefonů. Rozložení a funkčnost rozšířené aplikace byly testovány na různých zařízeních, aby byl zajištěn konzistentní a příjemný uživatelský zážitek bez ohledu na velikost obrazovky. Jednoduchost byla dalším klíčovým bodem rozšířené aplikace. Proběhla snaha minimalizovat zbytečné prvky a zjednodušit uživatelské rozhraní tak, aby uživatelé mohli snadno navigovat a rychle nalézt požadované informace. Uživatelé se tak mohou soustředit na hlavní účel aplikace a dosáhnout svých cílů efektivněji. Důraz byl kladen i na čitelnost informací pro uživatele. Byly použity srozumitelné texty a vhodné formátování pro zvýraznění důležitých informací, což umožňuje uživatelům snadno a rychle porozumět obsahům aplikace a lépe využít všechny její funkce.

Každá webová aplikace je neoddělitelně spojena s vlastními výzvami a požadavky, které vyžadují individuální přístup a pozornost. Během analýzy a tvorby aplikací byla využita široká škála zdrojů, včetně různých videí dostupných na YouTube, jež poskytla cenné návody a tipy od zkušených vývojářů. Díky nim bylo možné získat hlubší porozumění jednotlivým konceptům a technologiím, které jsou základem moderních frameworků. Použití různých dokumentací a online zdrojů také sehrálo klíčovou roli při pochopení funkcí a možností jednotlivých frameworků.

Je nutné zdůraznit, že každá webová aplikace by měla být připojena k databázi z důvodu uchovávání informací, jako je správa uživatelských účtů, registrovaných dat a dalších relevantních informací. I přesto, že připojení k databázi nebylo přímo implementováno v průběhu této práce, je to nezbytný krok pro budoucí vývoj a správné fungování webových aplikací. Tato doporučení jsou založena na důležitosti správného a bezpečného zpracování dat uživatelů, což je klíčový aspekt každé moderní webové aplikace. Práce poskytuje čtenářům ucelený pohled na moderní frameworky pro tvorbu webových aplikací, a to je klíčové pro zaujetí pozornosti na trhu. Díky porozumění principům web designu a responzivního designu, které jsou v práci zdůrazněny, čtenáři získají nástroje potřebné k vytvoření atraktivních a uživatelsky přívětivých webových aplikací. Tímto způsobem budou schopni oslovit širší publikum a zaujmout na konkurenčním trhu. Získané znalosti jim umožní vytvořit aplikace, které se pohodlně a esteticky přizpůsobí různým zařízením, což je klíčové pro úspěch v dnešní době, kdy uživatelé přistupují k webovým aplikacím z různých zařízení a obrazovek.

Pochopení všech různých frameworků vyžadovalo čas a úsilí věnované jejich studiu a porozumění. Je třeba zdůraznit, že popsat každý framework do detailů není úplně jednoduché, ale právě v této rozmanitosti a komplexnosti spočívá fascinace a výzva tvorby webových aplikací.

8 Závěry a doporučení

Práce nabídla analýzu současných frameworků pro vývoj webových aplikací a jejich praktické využití. Zhodnocení poskytlo porozumění technologiím, které formují digitální svět. S ohledem na neustálý pokrok v oblasti IT a rostoucí očekávání uživatelů je nutné se zaměřit na budoucí vývoj aplikací.

Vize do budoucna zahrnuje integraci pokročilých technologií, jako je umělá inteligence a strojové učení, do webových aplikací. Tyto technologie mohou přinést nové možnosti personalizace a interakce, což zlepší uživatelský zážitek a efektivitu aplikací. Dalším směrem je lepší využití rozšířené reality a virtuálních prostředí. Webové aplikace, jež využívají tuto technologii, mohou nabídnout uživatelům zcela nové způsoby interakce a prozkoumání digitálního světa.

Důležitým aspektem je také další zlepšení responzivity a přizpůsobivosti aplikací na různých zařízeních. Budoucnost patří mobilnímu prostředí a internetu věcí, a proto je nezbytné, aby byly webové aplikace optimalizované pro různé platformy a zařízení. Vedle technologických inovací je také důležité neustále zlepšovat uživatelské rozhraní a zážitek.

Implementace připojení k databázi, pravidelné testování aplikace na různých zařízeních a sledování nových trendů jsou klíčové pro udržení konkurenceschopnosti aplikace. Zapojení do komunitních aktivit a spolupráce s dalšími vývojáři může obohatit aplikaci o cenné poznatky a inspiraci. Vypracování plánu a strategie pro budoucí růst aplikace, který zahrnuje strategie pro získávání uživatelů a monetizaci aplikace, je klíčové pro dlouhodobý úspěch. Flexibilní a pružný plán umožní reagovat na změny a nové výzvy, jež se mohou v budoucnu objevit.

Respektování těchto doporučení může vést k úspěšnému rozvoji aplikace, která bude nejen atraktivní a užitečná pro uživatele, ale také konkurenceschopná na trhu dlouhodobě.

9 Seznam použitých obrázků

[39] ADOBE EXPERIENCE CLOUD TEAM, 2023. Single-page applications (SPAs) — what they are and how they work [online]. 19 July 2023. Available from: https://business.adobe.com/blog/basics/learn-the-benefits-of-single-page-apps-spa

[45] DOCKER INC., [no date]. Docker [online]. Available from: https://www.docker.com/resources/what-container/

10 Seznam použité literatury

[1] Ardalis, “Characteristics of modern web applications – .NET,” Microsoft Learn, Sep. 21, 2022. https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/modern-web-applications-characteristics

[2] A. Goel, “10 Best Web Development Frameworks to use in 2023,” Hackr.io. [Online]. Available: https://hackr.io/blog/web-development-frameworks

[3] M. Georgiou, “Enterprise Web App Design: 10 Key Principles that Lead to Better Usability in 2023,” Imaginovation | Top Web & Mobile App Development Company Raleigh. https://imaginovation.net/blog/enterprise-web-app-design-10-key-principles-better-usability-in-2021/

[4] W. Applications, “What’s the best way to use web application frameworks and libraries?,” www.linkedin.com, Aug. 30, 2023. https://www.linkedin.com/advice/3/whats-best-way-use-web-application-frameworks

[5] A. Kumar, “How modern web applications are made today - geek culture - medium,” Medium, Jan. 07, 2022. [Online]. Available: https://medium.com/geekculture/how-modern-web-applications-are-made-today-514ff5fc8506

[6] J. Patel and J. Patel, “Detailed Web framework comparison with features in 2023,” Monocubed, Sep. 15, 2023. https://www.monocubed.com/blog/web-development-framework-comparison/

[7] Coursera, “Marketing management: What is it and why does it matter?,” Coursera, Jun. 16, 2023. https://www.coursera.org/articles/marketing-management

[8] S. Pawar, “6 Key focus areas of Marketing Manager,” Jul. 23, 2018. https://www.linkedin.com/pulse/6-key-focus-areas-marketing-manager-shekhar-pawar

[9] K. Korn, “Sales vs. marketing: Which one should you focus on?,” Nov. 22, 2022. https://www.linkedin.com/pulse/sales-vs-marketing-which-one-should-you-focus-kristin-korn

[10] J. Joseph, “Why you need to focus on focus in your marketing plan,” Entrepreneur, Apr. 08, 2015. [Online]. Available: https://www.entrepreneur.com/growing-a-business/why-you-need-to-focus-on-focus-in-your-marketing-plan/244731

[11] M. Noorman and M. Noorman, “Here’s Where To Focus Your Marketing Efforts According to Experts,” Zen Media, May 15, 2023. https://zenmedia.com/blog/where-to-focus-marketing-efforts/

[12] “What is design thinking & why is it important? | HBS Online,” Business Insights Blog, Jan. 18, 2022. https://online.hbs.edu/blog/post/what-is-design-thinking

[13] A. Hayes, “Entrepreneur: What it means to be one and how to get started,” Investopedia, May 10, 2023. https://www.investopedia.com/terms/e/entrepreneur.asp

[14] D. H. Cto, “How to Make an App: Full Guide For 2023 | BuildFire - BuildFire,” BuildFire, Jul. 30, 2021. https://buildfire.com/how-to-create-a-mobile-app/

[15] “What is Design Thinking?,” The Interaction Design Foundation, Oct. 13, 2023. https://www.interaction-design.org/literature/topics/design-thinking

[16] “What is Web Design?,” The Interaction Design Foundation, Oct. 27, 2023. https://www.interaction-design.org/literature/topics/web-design

[17] “Website Design – The Leader in Website Design – Squarespace,” Squarespace. https://www.squarespace.com/website-design

[18] Angular, [no date]. [online]. Available from: https://angular.io/docs

[19] PATEL, Jeel and PATEL, Jeel, 2024. 5 Best Single Page Application Frameworks To Use in 2024. Monocubed [online]. 12 February 2024. Available from: https://www.monocubed.com/blog/top-single-page-application-frameworks/

[20] Introduction • Docs • Svelte, [no date]. [online]. Available from: https://svelte.dev/docs/introduction

[21] What Is a Single Page Application? | Bloomreach, [no date]. [online]. Available from: https://www.bloomreach.com/en/blog/2018/what-is-a-single-page-application

[22] Express - Node.js web application framework, [no date]. [online]. Available from: https://expressjs.com/

[23] SPOLEČNOST, Česká Astronomická, [no date]. Velká skvrna a silné erupce na Slunci. ČAS [online]. Available from: https://www.astro.cz/novinky/velka-skvrna-a-silne-erupce-na-slunci.html

[24] Čtk, [no date]. | ČeskéNoviny.cz. [online]. Available from: https://www.ceskenoviny.cz/zpravy/-bulharsko-a-rumunsko-jsou-po-13-letech-cekani-v-schengenskem-prostoru/2459801('

[25] KOUBOVÁ, Karolína and BREZOVSKÁ, Katarína, 2024. Rušení dalších poboček České pošty není na stole, říká manažer. Podle ekonoma je to ale třeba. iROZHLAS [online]. 1 April 2024. Available from: https://www.irozhlas.cz/zpravy-domov/ruseni-dalsich-pobocek-ceske-posty-neni-na-stole-rika-manazer-podle-ekonoma-je_2404011448_mst

[26] ČT24, Čtk, [no date]. Schodek rozpočtu dosáhl 105 miliard korun, pozitivně se projevila windfall tax. [online]. Available from: https://ct24.ceskatelevize.cz/clanek/ekonomika/rozpoctovy-deficit-byl-na-konci-prvniho-ctvrleti-105-miliard-v-breznu-se-mirne-zvetsil-347731

[27] PEŠEKOVÁ, Kamila, 2024. Slovensko vybíralo prezidenta, ale hlasovalo proti premiérovi Ficovi. iROZHLAS [online]. 26 March 2024. Available from: https://www.irozhlas.cz/komentare/slovensko-prezidentske-volby-pellegrini-korcok_2403260630_trs

[28] Robotizace pomáhá uchovat know-how | MM Průmyslové spektrum, [no date]. www.mmspektrum.com [online]. Available from: https://www.mmspektrum.com/technicke-novinky/robotizace-pomaha-uchovat-know-how

[29] Anarvin, 2024. Oscar 2024: Christopher Nolan ovládl ceny a zapsal se do filmové historie. FandímeFilmu.cz [online]. 3 November 2024. Available from: https://www.fandimefilmu.cz/clanek/34568-oscar-2024-christopher-nolan-ovladl-ceny-a-zapsal-se-do-filmove-historie

[30] POSKOČILOVÁ, Hanka, 2024. Hráči odevzdali maximum, ocenil po vyřazení hradecký trenér Tomáš Martinec. Hradecká Drbna - zprávy z Hradce a okolí [online]. 24 March 2024. Available from: https://hradecka.drbna.cz/sport/hokej/18320-na-lvy-zbyla-v-sezone-osma-pricka.html

[31] LEVINSON, Deborah a Todd BELTON. BUILD YOUR FIRST WEB APP: Learn to Build Web Applications from Scratch. 1. New York: Sterling Publishing, 2017, 280 s. ISBN 978-1-4549-2634-4.

[32] VOLLE, Adam. Web application. Encyclopedia Brittanica [online]. Chicago: Encyclopædia Britannica, 2022 [cit. 2023-01-13]. Dostupné z: https://www.britannica.com/topic/Web-application

[33] What Is a Web Application? (With Benefits and Jobs). Indeed Career Guide [online]. 2020. Available from: https://www.indeed.com/careeradvice/career-development/what-is-web-application

[34] What is a database?, [no date]. [online]. Available from: https://www.oracle.com/database/what-is-database/

[35] Database | Definition, Types, & Facts, 2024. Encyclopedia Britannica [online]. Available from: https://www.britannica.com/technology/database

[36] Ramotion, 2023. Best Practices to Develop Scalable Web Applications. Web Design, UI/UX, Branding, and App Development Blog [online]. 13 November 2023. Available from: https://www.ramotion.com/blog/scalable-web-applications/

[37] Ardalis, 2022. Characteristics of modern web applications - .NET. Microsoft Learn [online]. 21 September 2022. Available from: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/modern-web-applications-characteristics

[38] Ardalis, 2023. Choose between traditional web apps and single page apps - .NET. Microsoft Learn [online]. 25 February 2023. Available from: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps

[39] TEAM, Adobe Experience Cloud, [no date]. Single-page applications (SPAs) — what they are and how they work. [online]. Available from: https://business.adobe.com/blog/basics/learn-the-benefits-of-single-page-apps-spa

[40] Node.js — An introduction to the npm package manager, [no date]. [online]. Available from: https://nodejs.org/en/learn/getting-started/an-introduction-to-the-npm-package-manager

[41] ABBA, Ihechikara, 2024. What Is npm? An Introduction to Node's Package Manager. Kinsta® [online]. 12 March 2024. Available from: https://kinsta.com/knowledgebase/what-is-npm/

[42] MÁCA, Jindřich, [no date]. Lekce 1 - Úvod do Angular frameworku. [online]. Available from: https://www.itnetwork.cz/javascript/angular/zaklady/uvod-do-angular-frameworku

[43] React, [no date]. [online]. Available from: https://react.dev/

[44] Vue.js, [no date]. [online]. Available from: https://vuejs.org/guide/introduction

[45] What is a Container? | Docker, [no date]. Docker [online]. Available from: https://www.docker.com/resources/what-container/

[46] What is Java? - Java Programming Language Explained - AWS, [no date]. Amazon Web Services, Inc. [online]. Available from: https://aws.amazon.com/what-is/java/

[47] Learn Java - Dev.java, [no date]. Dev.java: The Destination for Java Developers [online]. Available from: https://dev.java/learn/

[48] Use Sass, Less, or CSS Within an Angular Component Template, [no date]. [online]. Available from: https://www.pluralsight.com/resources/blog/guides/use-sass-less-or-css-within-your-angular-component-template

11 Přílohy

A. Zdrojový kód pro Angular aplikaci
   https://github.com/Zeddrake/bp_prilohy/tree/main/Angular
B. Zdrojový kód pro React aplikaci
   https://github.com/Zeddrake/bp_prilohy/tree/main/React
C. Zdrojový kód pro Vue aplikaci
   https://github.com/Zeddrake/bp_prilohy/tree/main/Vue/todo-app
D. Zdrojový kód pro Svelte aplikaci
   https://github.com/Zeddrake/bp_prilohy/tree/main/Svelte/svelte-app

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**Ondřej Honců**
Studium:	I2000483
Studijní program:	B0688A140001 Informační management
Studijní obor:	Informační management
Název bakalářské práce:	**Moderní vývoj a design webových aplikací s využitím frameworků**
Název bakalářské práce AJ:	Modern web application development and design using frameworks

Cíl, metody, literatura, předpoklady:

Cílem práce je analýza moderních frameworků zaměřených na tvorbu a design webových aplikací, ukázána bude tvorba webových aplikací a stránek na moderních frameworcích s ukázkou kódu. Představení jejich základních principů, struktury a rozdílů mezi nimi. Následně bude provedeno rozšíření webové aplikace ve vybraném frameworku na modelovém příkladu.

Osnova

1. Úvod
2. Charakteristiky moderních webových aplikací
3. Jak zaujmout s webovou aplikací
4. Dostupnost moderních webových aplikací
5. Úvod do webdesignu
6. Single page framework
   - Angular
   - React
   - Vue
   - Svelte

7. Další frameworky pro vývoj webových stránek
   - Express
   - Java
8. Principy vývoje a návrhu moderních webových aplikací
9. Shrnutí výsledků
10. Závěry a doporučení

Literatura:
[1] LEVINSON, Deborah a Todd BELTON. BUILD YOUR FIRST WEB APP: Learn to Build Web Applications from Scratch. 1. New York: Sterling Publishing, 2017, 280 s. ISBN 978-1-4549-2634-4
[2] Ardalis, 2022. Characteristics of modern web applications - .NET. Microsoft Learn [online]. 21 September 2022. Available from: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/modern-web-applications-characteristics/
[3] GEORGIOU, Michael, [no date]. Enterprise Web App Design: 10 Key Principles that Lead to Better Usability in 2024. Imaginovation | Top Web & Mobile App Development Company Raleigh [online]. Available from: https://imaginovation.net/blog/usability-principles-enterprise-web-app-design/
[4] PATEL, Jeel and PATEL, Jeel, 2024. 5 Best Single Page Application Frameworks To Use in 2024. Monocubed [online]. 12 February 2024. Available from: https://www.monocubed.com/blog/top-single-page-application-frameworks/
[5] KUMAR, Ashutosh, 2022. How Modern Web Applications Are Made Today - Geek Culture - Medium. Medium [online]. 7 January 2022. Available from: https://medium.com/geekculture/how-modern-web-applications-are-made-today-514ff5fc8506/

Zadávající pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	Mgr. Hana Rohrová
Datum zadání závěrečné práce:	15.10.2021