Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Webové technologie

Semestrální projekt

Autor práce: Lukáš Hornych
Studijní obor: Aplikovaná Informatika

Vedoucí práce: Ing. Michal Macinka

Hradec Králové 2021

Anotace

Seminární práce se zabývá charakteristikou webových technologií a výběrem nejvhodnějších pro implementaci moderní webové aplikace umožňující vytváření a prohlížení profilových karet, která se svým chováním přibližuje nativním desktopovým a mobilním aplikacím. Práce nejprve popisuje dostupné webové technologie na trhu a jejich výhody a nevýhody a na závěr se věnuje výběru úzké skupině technologií splňující určité požadavky.

Klíčová slova

web, technologie, server, klient, backend, frontend, server-side, client-side

Obsah

1 Úvod

V dnešním moderním světě existuje spousta webových technologií, které umožňují a většinou i značně usnadňují vývoj webových stránek, aplikací či mikroslužeb. Nové alternativní technologie každým dnem přibývají a může být mnohdy obtížné i pro zkušené vývojáře se zorientovat a vybrat tu správnou pro daný projekt. Při rozhodování je důležité vybírat nejen podle popularity, ale především podle typu projektu, ten totiž může zásadně ovlivnit, jaké technologie jsou vhodné a které nikoliv.

1.1 Cíl seminární práce

Cílem této seminární práce je vysvětlit základní pojmy a rozdělení webových technologií a následně charakterizovat ty nejpoužívanější. Součástí charakteristiky každé z nich bude seznámení a shrnutí kladů a záporů v porovnání s konkurencí. V závěru bude následovat stanovení kritérií pro výběr nejvhodnější technologie a samotný výběr.

2 Webové technologie

Webové technologie se primárně rozdělují na dvě kategorie: front-endové a back-endové. Obě obsahují rozdílné technologie zejména kvůli tomu, že každá každá z nich zabývá odlišnou částí vývoje. Existují ale i technologie spadající více či méně do obou kategorií, avšak v každé plní trochu jiný účel. Nejrozšířenějším takovým kandidátem je JavaScript (JS) dominující v současné době oběma kategoriím.

2.1 Front-endové technologie

Front-endové technologie se věnují především přímé interakci s uživatelem pomocí webového prohlížeče. Definují tak vzhled a samotné ovládání aplikace pro koncového uživatele. Tyto technologie pak zpravidla jaksi obalují a zpracovávají funkcionality serverů tak, aby se koncovému uživateli s danými daty pracovalo co možná nejjednodušeji a nebyl nucen pracovat se surovými daty. V drtivé většině pak neposkytují uživatelům jen data samotná, ale hlavně informace z nich zpracované.

Tyto technologie by se daly ještě pomyslně rozdělit na standardizované a ostatní. Standardizované tvoří základní stavební kameny všech webových stránek a aplikací, kterým rozumí webové prohlížeče. Ostatní pak představují jakési nadstavby nad těmi standardizovanými a snaží se je nějakým způsobem rozšířit či zjednodušit. Tyto už standardizované nijak nejsou a mohou se proto často zásadním způsobem měnit od verze k verzi. Je jich mnohem více a každým dnem vznikají nové, vývojářům značně ulehčují práci, a proto jsou tolik oblíbené.

2.1.1 Standardizované front-endové technologie

Jsou definovány standardy společností W3C, což umožňuje vyvíjet v podstatě univerzální stránky a aplikace běžící na koncovým uživatelem zvoleném prohlížeči. Zástupci těchto technologií jsou HTML, CSS, JS a nově i WebAssembly a představují tedy základní nástroje pro tvorbu jakýchkoliv stránek a aplikací jež je možné provozovat na internetu. [1]

Nejdůležitější je značkovací jazyk HyperText Markup Language (HTML), bez kterého se nelze obejít. Tento jazyk definuje základní strukturu a obsah pomocí tzv. značek, které se do sebe zanořují a tvoří tak stromovou strukturu bloků dokumentu. Struktura vychází z jazyka Extensible Markup Language (XML), který je obohacen hlavně o vlastní značky s přidaným významem pro orientaci prohlížečů při vykreslování stránek. [2]

Další velmi důležitou technologií je stylovací jazyk Cascading Style Sheets (CSS) definující vzhled struktury dokumentu pro koncové uživatele. Pomocí tohoto jazyka je možné upravovat písma, barvy, pozice prvků dokumentu a dokonce i animace. Vzhled stránek a aplikací je čím dál tím důležitější jak pro snazší orientaci, tak hlavně pro odlišení se od konkurence. [3]

V dnešní době už téměř stejně důležitý stavební prvek jako HTML nebo CSS je skriptovací jazyk JS umožňující na front-endu dynamicky, v průběhu interakce uživatele se stránkou, modifikovat strukturu a vzhled původní stránky či aplikace. To otevírá obrovské možnosti pro tvorbu komplexních animací, dynamického donačítání obsahu ze serveru, her, přehrávačů videí a mnoho dalšího. [4]

Alternativou případně doplňkem k JS je nový standard WebAssembly. Je to nízkoúrovňový jazyk podobný Assembleru sloužící jako univerzální jazyk, do kterého je možné překládat kód z mnoha již existujících jazyků jako např. C++ nebo Rust. Oproti JS má ohromnou výkonnostní výhodu, což otevírá možnost pro výpočetně náročné aplikace běžet

v klasickém prohlížeči. WebAssembly se ale dá použít společně s JS a lze tak využít výhody obou jazyků. [5]

2.1.2 CSS preprocesory, frameworky a knihovny

Používání základního jazyka CSS může být zejména u větších projektů mnohdy těžkopádné a špatně udržovatelné. Proto existují nástroje jako preprocesory, frameworky a knihovny ulehčující zásadním způsobem práci vývojářům.

Preprocesory jsou jazyky představující jakousi nadstavbu základního CSS a rozšiřují ho o vlastní funkcionality. Problém je v tom, že prohlížeče rozumí pouze standardizovanému CSS, proto musí být tyto jazyky překládány zpět do CSS, který lze pak použít standardním způsobem.

Frameworky a knihovny pak seskupují předpřipravené kusy CSS kódu připravené pro rychlé použití při tvorbě stránek. Díky nim je možné rychle stylovat dokument stránky, protože vývojář nemusí pomocí CSS konfigurovat vše, co se týče vzhledu, ručně. Místo toho může použít výše zmíněné kusy kódů nebo již celé hotové bloky.

Nejrozšířenějším preprocesorem je jednoznačně Syntactically Awesome Style Sheets (Sass). Oproti základnímu CSS totiž nabízí širokou škálu funkcionalit od vnořování, funkcí pro automatizaci až po třeba dědičnost. Prohlížeče ale tomuto jazyku nerozumí, proto je nutné ho překládat do CSS, pomocí předpřipravených nástrojů. Tento jazyk mimo jiné hlavně usnadňuje organizaci stylovacího kódu a je tak jednodušší ho v budoucnu rozšiřovat a modifikovat. [6]

Méně používanou alternativou k Sass je Leaner Style Sheets (Less) a stejně jako Sass rozšiřuje základní CSS o další funkcionality jako vnořování nebo dědičnost a je též nutné ho překládat do standardizovaného CSS. [7] Less ale oproti Sass ve spoustě věcech pokulhává a proto není mezi vývojáři tolik oblíbený. Sass má např. prokročilejší možnosti scriptování zahrnující cykly a podmínky blížící se programovácím jazykům a umožňuje tak jakousi automatizaci generovaného CSS kódu. Naproti tomu Less poskytuje jen základní cykly a proměnné což může být pro některé vývojáře dosti omezující. Sass má také mnohem pokročilejší a efektivnější dědičnost, nastavení neduplikuje ale spíše nahrazuje. [8].

Dlouholetým favoritem mezi frameworky je Bootstrap. Ten obsahuje velké množství základních předpřipravených šablon, nastavení typografie, formulářů, tlačítek atd. Kromě toho lze využít komunitních šablon celých stránek a vytvořit finální stránky během velmi krátké doby. Bootstrap je ale paměťově náročný a mnohdy těžkopádný, a proto se ho, pokud využívají jen část jeho funkcionalit, někteří vývojáři zbavují, a nahrazují ho jednoduššími alternativami či přímo nově vznikajícími standardy v samotném CSS. [9]

Poměrně novou oblíbenou alternativou k Bootstrapu je TailwindCSS poskytující obecné CSS třídy pro pozicování a stylování HTML struktury. To umožňuje, stejně jako u Bootstrapu, rychlou tvorbu webových stránek a aplikací bez nutnosti hlubších znalostí samotného jazyka CSS. [10] Oproti Bootstrapu je ale mnohem univerzálnější a díky univerzálním třídám, stránky nevypadají podobně jako v případě Bootstrapu, který poskytuje spíše předpřipravené bloky či rovnou celé stránky. TailwindCSS je také méně paměťově náročný, a proto nepředstavuje tak velkou zátěž jako Bootstrap. [11]

2.1.3 JavaScript frameworky a knihovny

Frameworky a knihovny představují v JS sadu předpřipravených nástrojů pro snazší a rychlejší vývoj oproti čistému JS. Dříve vývojáři hojně využívali knihoven pro chybějící základní funkcionality, protože nebyl v některých ohledech tak pokročilý. Dnes je čistý JS velmi po-

kročilý a není problém používat pouze něj, avšak dnes se spíše používají frameworky a knihovny pro snadnou tvorbu reaktivních znovupoužitelných komponent. Takovéto komponenty představují samostatné bloky, jako např. formuláře nebo tlačítka, udržující si vlastní data a při změně se automaticky překreslují.

Momentálně nejznámější knihovnou pro tvorbu takových komponent je React, jenž byl vytvořen především pro tvorbu interaktivních grafických uživatelských rozhraní (GUI) webových stránek a aplikací. Zaměřuje se hlavně na zobrazovací vrstvu, což zahrnuje reaktivnost a překreslování komponent a skládání výsledné stránky z takových komponent. Ostatní funkcionality pak spíše přenechává již existujícím specializovaným knihovnám, které jsou často lepší kvůli delšímu vývoji a velké komunitě. Stejně jako další podobné frameworky je postaven na systému reaktivity, kdy každá komponenta má vlastní data, která když se změní, část stránky se automaticky překreslí s novými daty bez nutnosti explicitního překreslení vývojářem. Nicméně React může být složitější na naučení a pochopení, kvůli horší dokumentaci nebo rychlému vývoji. Rychlý vývoj tohoto frameworku zesložiťuje i samotný vývoj projektů, protože se spoustu věcí často mění a existující projekty mohou mít problémy s pozdějšími upgrady na novější verze. Má ale velkou komunitu a s tím i spojené velké množství materiálů. Navíc je vyvíjen samotným Facebookem, což zaručuje, že se bude ještě dlouhou dobu rozvíjet a jen tak nezmizí. [12]

Další čím dál více oblíbenou alternativou pro snadnou tvorbu interaktivních GUI je Vue. Stejně jako React se zaměřuje pouze na zobrazovací vrstvu s využitím reaktivních komponent a ostatní funkce přenechává již existujícím knihovnám. To umožňuje vývojářům vybrat správné technologie pro daný projekt a nemuset se omezovat vybraným hlavním frameworkem. Jeho velkou výhodou je flexibilita míry integrovanosti do projektu. Lze ho totiž použít buď jako pomocníka při tvorbě jednoduchých komponent ve stávajícím projektu, nebo jako celou platformu pro tvorbu sofistikovaných webových aplikací. [13] Vue je oproti Reactu mnohem jednodušší na naučení, jak pro zkušené vývojáře, tak pro začátečníky s programováním, a společně s ne tak často měnícím se kódem je čím dál více používán nejen jednotlivci, ale i velkými firmami. Nemá sice zatím tak velkou komunitu, ale je dostatečně velká na to, aby se o něm dalo uvažovat jako o solidní možnosti pro nový projekt. [14]

Svelte je nejnovějším přírůstkem do skupiny nejpoužívanějších frameworků pro tvorbu GUI, avšak přináší odlišný způsob tvorby interaktivních stránek a aplikací. Stejně jako Vue je možné ho použít jen okrajově jako pomocníka nebo pomocí něj vytvořit celou aplikaci. Od konkurence React a Vue se ale zásadně liší ve formě v jaké běží v prohlížeči. Zatím co React a Vue používají svojí logiku za chodu aplikace k sestavování stránek, Svelte překládá kód do čistého kompaktního JS kódu. To má vliv především na rychlost celé stránky či aplikace, především pak u složitých aplikací, ovšem za cenu vyšší rychlosti je zde omezení v podobě vlastního scriptovacího jazyka. [15] Ten se podobá klasickému JS, ale nelze říct, že je s ním zaměnitelný. V případě potíží při vývoji tak může být pro některé vývojáře obtížné danou situaci vyřešit. [16]

Mezi tyto frameworky by dal zařadit ještě Angular, který už však ztrácí na popularitě a je používán spíše v korporátní sféře. Angular je spolehlivý a mocný nástroj a poskytuje obdobné funkcionality jako konkurence, nicméně nemá moc dobrou dokumentaci a je obtížný na naučení. [17]

2.2 Back-endové technologie

Back-endové technologie běží na serverech a proto se na rozdíl od front-endových vůbec nedostanou do styku s koncovými uživateli. Místo toho se zaměřují hlavně na práci s daty

a informacemi v podobě poskytování relevantních dat front-endovým technologiím a provádění mnohdy výpočetně náročných operací nad uživatelskými daty.

Pro komunikaci mezi těmito dvěma světy se používá standardizovaný aplikační protokol Hypertext Transfer Protocol (HTTP). Ten definuje strukturu přenášených dat, tak aby jakýkoliv server s jakoukoliv technologií mohl bez problému komunikovat standardizovaným způsobem s jakýmkoliv prohlížečem. [18]

Díky tomuto protokolu servery nejsou omezeny základními standardizovanými technologiemi jako tomu je v případě HTML, CSS a JS, a vývojáři tak mohou používat teoreticky jakýkoliv programovací jazyk a ekosystém s ním spojený.

2.2.1 Java

Java je velmi univerzální objektově orientovaný programovací jazyk, jenž může být provozován téměř na jakémkoliv hardwaru od serveru až po mikrořadiče. Umožňuje vyvíjet ohromnou škálu aplikací počínaje desktopovými aplikacemi, přes hry až po třeba právě ty serverové aplikace. [19] Ačkoliv je psaní kódu v Javě oproti jiným jazykům mnohdy zbytečně zdlouhavé, které nabízejí kratší zápisy některých příkazů, Java nabízí velmi stabilní a zpětně kompatibilní prostředí pro vývoj. Není se proto třeba obávat, že by se v příští verzi udály velké zpětně nekompatibilní změny nebo dokonce by přestala být zcela podporována. Java se osvědčila jako vhodná i pro výpočetně náročné serverové aplikace potřebující provádět několik paralelních operací najednou.

V dnešní době jsou serverové aplikace čím dál tím složitější a je v podstatě nutné používat nějaký framework či knihovnu, poskytující nástroje pro práci s protokolem HTTP a všeho s tím spojené, pro udržení přehledného a snadno rozšiřitelného kódu.

Jedním z nejpoužívanějších frameworků pro vývoj webových aplikací v Javě je Spring. Věnuje se velké škále různých nástrojů pro vývoj a mezi ty hlavní patří systém Dependency injection (DI) usnadňující práci s mnoho objekty, které jsou mezi sebou propojené a bylo by obtížné tyto vazby spravovat svépomoci. Neméně důležitou funkcionalitou je systém událostí poskytující vývojářům možnost jednoduše v rámci programu vyvolávat události, na které se lze kdekoliv v kódu napojit a spouštět potřebné procesy. [20] Pro vývoj webových stránek pak poskytuje nástroje pro implementaci Model-view-controller (MVC) architektury, Representational State Transfer (REST) API architektury atp. [21] MVC architektura rozděluje zpracovávání požadavku mezi datový model starající se o data a jejich přípravu, pohled sestavují výsledné stránky s připravených dat a v poslední řadě řadič zpracovávající požadavky, které následně deleguje na datový model a výsledná data předá pohledu. [22] Čím dál více využívanou architekturou při tvorbě webů je REST API. Ta se vůbec nestará o to jak se data dostanou ke koncovému uživateli natož v jaké podobě. Místo toho se soustředí pouze na zpracování požadavku a poskytnutí relevantních dat, se kterými pak většinou pracují front-endové technologie pro sestavení stránek a aplikací. [23] Nicméně Spring poskytuje velké množství dalších nástrojů a funkcionalit, a proto může být pro začínající vývojáře jeho rozsáhlost odrazující. Avšak znalost tohoto frameworku se značně vyplatí, hlavně pak při tvorbě velkých projektů; příkladem mohou být třeba e-shopy.

Alternativou k Springu je Java EE resp. v současné době Jakarta EE (pouze jiný název). Ta definuje oficiální standardy pro různé způsoby zpracování HTTP požadavků od nejzákladnějšího obecného zpracování, až po např. formátování odpovědi podle daných požadavků. Zároveň definuje standardy např. pro mapování objektů na databázové tabulky, DI a atp. Tím, že se jedná ve své podstatě pouze o standardy je zapotřebí vybrat pro vývoj knihovny implementující právě tyto standardy. [24]

Obě technologie jsou velmi populární a velmi mocné. Každá z nich má své klady a zápory a nelze proto jednoznačně určit lepší z nich, nicméně Spring je obecně jednodušší při vývoji a navíc je zdarma. Java EE naproti tomu přichází s Oracle licencí, a proto se více hodí pro použití ve velkých korporátních společnostech. [25]

2.2.2 Javascript

V dnešní době je možné scriptovací jazyk JS používat nejen k vývoji interaktivního GUI, ale také k vývoji serverových aplikací zpracovávající HTTP požadavky za pomoci běhového prostředí Node.js využívající jádro V8, stejně jako prohlížeče. [26] Ačkoliv JS běží pouze jednovláknově, je poměrně výkonný pro obsluhu velkého množství menších dotazů díky systému událostí, kdy se zpracovává vždy jen ten nejdůležitější požadavek. Není ale úplně vhodný pro výpočetně náročné operace, protože pak může blokovat ostatní požadavky, právě kvůli neschopnosti rozdělit požadavky mezi více vláken. [27] Pro některé vývojáře může být použití výhodou, protože v případě použití JS též na front-endu, nemusí řešit rozdílné jazyky a knihovny. Jednoduše použijí jeden jazyk a podobné knihovny. Stejně jako v případě Javy existuje mnoho frameworků a knihoven poskytující předpřipravené nástroje pro ulehčení tvorby webových stránek.

Momentálně nejpoužívanějším frameworkem pro back-endový JS běžící na Node.js je Express.js. [28] Ten umožňuje především přijímat samotné HTTP požadavky a odesílat HTTP odpovědi a také poskytuje vývojářům prostředky pro zpracovávání těchto požadavků pomocí architektury MVC, REST API atd., podobně jako Spring v Javě. Mimo jiné je pro spoustu ostatních frameworků základním stavebním kamenem, ke kterému pak přidávají své funkcionality. [26]

Next.js je rovněž frameworkem běžícím na Node.js, ale cílí na front-endovou knihovnu React. Jeho hlavním úkolem je zjednodušení jeho provozu v produkčním prostředí. Může být totiž obtížné nastavit prostředí serveru pro správné poskytování Reactu pro front-end. Kromě výše zmíněného jej rozšiřuje např. o schopnost vykreslení částí nebo celků stránek již na serveru a prohlížeči posílá připravenou stránku či aplikaci. [29]

Obdobou Next.js pro front-endový Vue je Nuxt.js. Ten poskytuje dost podobné funkcionality jako usnadnění provozu Vue v produkčním prostředí, sestavování částí nebo celků stránek nebo třeba jednodušší podporu pro navigaci mezi jednotlivými stránkami. [30]

Při výběru mezi Nuxt.js a Next.js tak záleží především na volbě front-endového frameworku či knihovny nikoliv obráceně, protože nelze použít Nuxt.js společně s Reactem a naopak.

2.2.3 PHP

Dalším stále velmi oblíbeným, i přes jeho stáří, je skriptovací jazyk PHP: Hypertext preprocessor (PHP) zaměřující se především na tvorbu webových stránek a aplikací. [31]

Tento jazyk je již sám o sobě mocný a dokáže spoustu věcí. Avšak poskytuje spíše základní nástroje a vývojáři musí tak psát pokročilejší logiku pokaždé od nuly. Proto existují frameworky poskytující jistou úroveň abstrakce vzdávající vývojáře nutnosti tvorby základní logiky, jako navigaci mezi stránkami, obsluhu požadavků atp., která je mnohdy pro většinu stránek a aplikací podobná né-li stejná. [32]

V České Republice je velmi oblíbený český framework Nette díky jeho přehlednosti a univerzálnosti. Poskytuje širokou škálu nástrojů a funkcionalit podobně jako Spring pro Javu. Též umožňuje vývojářům využít hotového systému DI, implementovat MVC archi-

tekturu nebo třeba používat vlastní šablonovací systém pro tvorbu zobrazovací vrstvy v rámci MVC. [33]

Na poli globálně populárních frameworků dominuje především Laravel. Mimo základní funkce přináší schopnost stavět komplexní webové aplikace s velkou rychlostí a bezpečností v kombinaci s jednoduchostí vývoje, kterou přinášejí předpřipravené nástroje starající se o běžné repetitivní úkoly. Kvůli jeho velké popularitě přichází s velkým ekosystémem v podobě např. dostupnosti hostujících serverů s instalací na jedno kliknutí nebo velkého množství návodů. [32]

Největší konkurencí Laravelu je Symphony, který je starší a má podobně velkou komunitu i ekosystém. Narozdíl od Laravelu je rozdělen do komponent podle zaměření funkcionalit a je tak možné je používat samostatně. [32] Symfony oproti Laravelu poskytuje nástroje pro velké projekty jako např. škálování. Nicméně se může jevit jako složitější na pochopení pro začínající vývojáře a může být výrazně pomalejší než Laravel, kvůli externím knihovnám. Naproti tomu Laravel se často mění a může být proto obtížné stávající projekty aktualizovat. [34]

2.2.4 C#

Podobně jako Java je C# univerzální objektově orientovaný programovací jazyk pro tvorbu nejen desktopových aplikací a her, ale právě i webových stránek a aplikací. K tvorbě webových stránek existuje oficiální framework ASP.NET, který rozšiřuje základní schopnosti komunikace pomocí HTTP protokolu o pomocné nástroje pro snadný vývoj webových aplikací, REST API nebo třeba komunikace mezi prohlížečem a serverem v reálném čase. [35]

Zajímavou a nadějnou novinkou v rámci frameworku ASP.NET je Blazor. Ten umožňuje vývojářům vyvíjet i front-endovou interakci s koncovým uživatelem pomocí pouze C#, HTML a CSS bez jakékoliv nutnosti znát JS, přičemž Blazor nabízí dvě možnosti, jak tohoto docílit; a to pomocí WebAssembly, nebo pomocí již zmíněné komunikace mezi prohlížečem a serverem v reálném čase. V případě WebAssembly se kód napsaný v C# překládá do právě WebAssembly a běží tak čistě v prohlížeči jako front-endová technologie. Pokud si ale vývojář vybere druhou možnost, v prohlížeči běží jen malý komunikační nástroj delegující operace spojené s interaktivním front-endem na server, kde je kód vykonáván přímo v C#. Obě možnosti pak dovolují používat již existující knihovny pro .NET ekosystém, což může být pro některé vývojáře obrovskou výhodou stejně jako fakt, že nemusí programovat v JS. Nicméně vzhledem k tomu, že je tato technologie poměrně nová, nemusí být pro všechny projekty vhodné spoléhat se na nezaběhlou a neověřenou technologii, protože není jasné jestli budu ještě dlouhé roky podporována a navíc nemá z daleka tak velkou komunitu jako např. React nebo Vue. [36]

3 Závěr

Je patrné, že vybrat správnou technologii pro daný projekt není snadný úkol, především pokud vývojář nemá ani okrajové zkušenosti s většinou těchto technologií a nemůže tedy vybírat podle vlastních poznatků. Pokud jsou ale vybrány celosvětově používané a podporované technologie, nelze udělat chybu výběrem ani jednou z nich, protože každá má své výhody. Hlavní rozdíl pro většinu vývojářů spočívá ve stylu, jakým se v dané technologii vyvíjí a jestli daný vývojář má alespoň nějaké znalosti používaného programovacího či skriptovacího jazyka.

3.1 Kritéria pro výběr technologií

Vyvíjený projekt se bude zabývat možností tvorby a zveřejňování profilových karet obsahující např. odkazy na sociální sítě, kontaktní údaje nebo popis daného subjektu.

Hlavním požadavkem je vytvořit webovou aplikaci, která se bude chováním a interakcí co nejvíce přibližovat k nativní aplikaci jak pro desktop, tak pro mobilní aplikaci zároveň (podle typu zařízení, na kterém aplikace v danou chvíli poběží) bez nutnosti instalace takové aplikace, aby byl poskytnut co možná nejrychlejší přístup k informacím v dané aplikaci.

Dalším velmi důležitým aspektem pro aplikaci poskytující spoustu vyhledatelných informací je možnost vytvořit kvalitní SEO, aby se daná aplikace dostala do předních žebříčků při vyhledávání uživateli skrze internetové vyhledávače. Search engine optimization (SEO) je tedy technika pro optimalizaci webových stránek pro internetové vyhledávače jako např. Google, Bing nebo DuckDuckGo, tak aby byly schopné co nejlépe pochopit a kategorizovat informace poskytnuté danou stránkou a předat dané informace koncovému uživateli. [37]

3.2 Výběr technologií

Vzhledem k tomu, že cílem je vytvořit webovou aplikaci, která poběží v standardních webových prohlížečích, a tím pádem se nelze vyhnout základním front-endovým technologiím HTML, CSS a JS. Ty lze ale do jisté míry nahradit jejich nadstavbami.

V případě CSS byl vybrán preprocesor Sass, pro jeho širokou komunitu a funkce. Kvůli výše zmíněné potřebě zobrazovat různé typy odkazů na profilových kartách, Sass umožní do jisté míry automatizovat repetitivní a nepřehledné CSS třídy.

V případě tvorby interaktivního GUI je výběr složitější. Je možno využít jak základního JS, tak frameworků či knihoven usnadňující tvorbu reaktivních komponent. Alternativou může být jazyk C# s pomocí technologie Blazor, díky kterému je možné se úplně od JS odstínit. Ze studijních účelů byl vybrán JS, především pro hlubší pochopení tohoto jazyka, protože je používán v současné době téměř každou webovou stránkou či aplikací a pravděpodobně tomu tak ještě dlouhou dobu bude. Jako nadstavba jazyka pro zrychlení vývoje byl vybrán framework Vue díky jeho velmi dobré dokumentaci, snazšího pochopení a možnosti programovat přímo v jazyku JS. Vue (a stejně tak ostatní takové nadstavby) sám o sobě ale nedisponuje zrovna dobrými prostředky pro SEO, což je dáno delegací sestavení výsledné stránky až na front-endový JS. Kvůli tomu vyhledávače při strojovém procházení takových stránek nevidí výslednou stránku a nemohou tak vyextrahovat potřebné informace. Řešením může být vykreslení takové stránky již na serveru a prohlížeči poslat již připravenou stránku, kterou už následně převezme samotný Vue. Tuto techniku poskytují již hotovou např. serverové frameworky Nuxt.js nebo Next.js. Protože byl ale vybrán Vue, lze vybrat pouze Nuxt.js mající tuto techniku jako jednu z jeho hlavních výhod.

Co se týče serverových technologií, tam je výběr, v případě aplikace či stránky bez nutnosti výpočetně náročných operací, spíše otázka preferencí a předchozích znalostí daného vývojáře. Tím, že byl již vybrán serverový framework Nuxt.js pro podporu SEO, by bylo možné jednoduše použít Node.js, na kterém Nuxt.js staví v kombinaci např. s frameworkem Express.js a vystavit REST API přímo pomocí něj. Ekvivalentní alternativou pro tvorbu samotného REST API je použití např. jazyka Java, C#, JS nebo PHP. Ze studijních důvodů a vývojářovy předchozí znalosti byl vybrán právě jazyk Java společně s frameworkem Spring poskytující všechny nástroje potřebné pro jednoduchou tvorbu REST API a jeho automatizované testování bez nutnosti řešit licenční poplatky jako v případě Java EE.

Literatura

[1] W3C. Web Design and Applications. 2016, [Online; access 12. 08. 2021]. Available from: https://www.w3.org/standards/webdesign/

[2] MDN contributors. HTML: HyperText Markup Language. 2021, [Online; access 11. 08. 2021]. Available from: https://developer.mozilla.org/en-US/docs/Web/HTML

[3] MDN contributors. CSS: Cascading Style Sheets. 2021, [Online; access 11. 08. 2021]. Available from: https://developer.mozilla.org/en-US/docs/Web/CSS

[4] MDN contributors. What is JavaScript. year, [Online; access 11. 08. 2021]. Available from: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[5] MDN contributors. WebAssembly. 2021, [Online; access 12. 08. 2021]. Available from: https://developer.mozilla.org/en-US/docs/WebAssembly

[6] Sass team. Sass Basics. 2021, [Online; access 11. 08. 2021]. Available from: https://sass-lang.com/guide

[7] Less team. Less Overview. 2021, [Online; access 11. 08. 2021]. Available from: https://lesscss.org/

[8] Coyier, C. Sass vs. Less. 2012, [Online; access 11. 08. 2021]. Available from: https://css-tricks.com/sass-vs-less/

[9] Wikipedia contributors. Bootstrap (front-end framework) — Wikipedia, The Free Encyclopedia. 2021, [Online; access 11. 08. 2021]. Available from: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

[10] TailwindCSS. TailwindCSS. [Online; access 11. 08. 2021]. Available from: https://tailwindcss.com/

[11] Mourya, N. Tailwind CSS vs. Bootstrap: Which Is a Better Framework? 2021, [Online; access 11. 08. 2021]. Available from: https://www.makeuseof.com/tailwind-css-vs-bootstrap-which-is-a-better-framework/

[12] Facebook. React. 2021, [Online; access 11. 08. 2021]. Available from: https://reactjs.org/

[13] Vue contributors. Vue Guide. 2021, [Online; access 11. 08. 2021]. Available from: https://vuejs.org/v2/guide/

[14] Mistry, J. Vue vs. React. 2021, [Online; access 11. 08. 2021]. Available from: https://www.monocubed.com/vue-vs-react/

[15] Svelte contributors. Svelte Basics. 2019, [Online; access 11. 08. 2021]. Available from: https://svelte.dev/tutorial/basics

[16] Li, A. Vue vs Svelte: Comparing Framework Internals. 2021, [Online; access 11. 08. 2021]. Available from: https://www.vuemastery.com/blog/vue-vs-svelte-comparing-framework-internals/

[17] Technostacks. React vs Angular: Which Is A Better JavaScript Framework? 2020, [Online; access 12. 08. 2021]. Available from: https://technostacks.com/blog/react-vs-angular

[18] MDN contributors. HTTP. 2021, [Online; access 13. 08. 2021]. Available from: https://developer.mozilla.org/en-US/docs/Web/HTTP

[19] Wikipedia contributors. Java (programming language) — Wikipedia, The Free Encyclopedia. 2021, [Online; access 13. 08. 2021]. Available from: https://en.wikipedia.org/wiki/Java_(programming_language)

[20] Spring. Spring Framework Documentation Core. 2021, [Online; access 14. 08. 2021]. Available from: https://docs.spring.io/spring-framework/docs/current/reference/html/core.html

[21] Spring. Spring Framework Documentation Web. 2021, [Online; access 14. 08. 2021]. Available from: https://docs.spring.io/spring-framework/docs/current/reference/html/web.html

[22] Wikipedia contributors. Model–view–controller. 2021, [Online; access 14. 08. 2021]. Available from: Model-view-controller---Wikipedie:Otevřenáencyklopedie

[23] restfulapi.net. What is REST. 2020, [Online; access 14. 08. 2021]. Available from: https://restfulapi.net/

[24] Wikipedia contributors. Jakarta EE — Wikipedia, The Free Encyclopedia. 2021, [Online; accessed 15. 08. 2021]. Available from: https://en.wikipedia.org/w/index.php?title=Jakarta_EE&oldid=1034427545

[25] Xperti. Java EE vs. Spring: Which is more popular? 2021, [Online; accessed 15. 08. 2021]. Available from: https://xperti.io/blogs/java-ee-vs-spring/

[26] MDN contributors. Express/Node introduction. 2021, [Online; access 14. 08. 2021]. Available from: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

[27] MDN contributors. Concurrency model and the event loop. 2021, [Online; accessed 15. 08. 2021]. Available from: https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop

[28] Nivo. State Of JavaScript 2020. 2020, [Online; access 14. 08. 2021]. Available from: https://2020.stateofjs.com/en-US/technologies/

[29] Vercel. Create a Next.js App. 2021, [Online; access 14. 08. 2021]. Available from: https://nextjs.org/learn/basics/create-nextjs-app

[30] Nuxt.js contributors. Nuxt.js. 2021, [Online; access 14. 08. 2021]. Available from: https://github.com/nuxt/nuxt.js

[31] Wikipedia contributors. PHP — Wikipedia, The Free Encyclopedia. 2021, [Online; access 14. 08. 2021]. Available from: https://en.wikipedia.org/wiki/PHP

[32] Bhatia, S. Best PHP Frameworks for Web Development. 2021, [Online; access 14. 08. 2021]. Available from: https://hackr.io/blog/best-php-frameworks

[33] Nette Foundation. Proč používat Nette. 2021, [Online; access 14. 08. 2021]. Available from: https://doc.nette.org/cs/3.1/why-use-nette

[34] Asper Brothers. Laravel vs Symfony in 2021. 2019, [Online; access 14. 08. 2021]. Available from: https://asperbrothers.com/blog/laravel-vs-symfony/

[35] Microsoft. ASP.NET. 2021, [Online; access 15. 08. 2021]. Available from: https://dotnet.microsoft.com/apps/aspnet

[36] Microsoft. Blazor. 2021, [Online; access 15. 08. 2021]. Available from: https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor

[37] Thirt Door Media. What Is SEO / Search Engine Optimization? 2021, [Online; access 15. 08. 2021]. Available from: https://searchengineland.com/guide/what-is-seo

Seznam zkratek

CSS Cascading Style Sheets

DI dependency injection

GUI grafické uživatelské rozhraní

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

JS JavaScript

Less Leaner Style Sheets

MVC Model-view-controller

PHP PHP: Hypertext preprocessor

REST Representational State Transfer

Sass Syntactically Awesome Style Sheets

SEO search engine optimization

XML Extensible Markup Language