UNIVERZITA PALACKÉHO V OLOMOUCI

PEDAGOGICKÁ FAKULTA

Katedra technické a informační výchovy

Bakalářská práce

David Grydil

Využití webových technologií pro tvorbu moderního webu

Olomouc 2023 vedoucí práce: prof. PhDr. Milan Klement Ph.D.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pouze s využitím uvedených pramenů a literatury.

V Olomouci 15. 6. 2023 ……..……..

David Grydil

Chtěl bych poděkovat prof. PhDr. Milanu Klementovi Ph.D. za to že se ujal vedení mé práce, za jeho cenné rady a poznámky při konzultacích ale i mimo ně.

Všechny překlady v této práci byly vytvořeny autorem této práce.

Obsah

Úvod

Rychlý rozvoj webových technologií změnil digitální prostředí a dal vzniknout modernímu webu. V této práci se zabýváme základními prvky a vybranými populárními technologiemi, které pomohly při revoluci ve vývoji webových stránek a uživatelských zkušenostech.

Jádrem moderního webu je bezproblémová integrace HTML5, CSS3 a JavaScriptu. Tyto základní technologie představují stavební kameny pro tvorbu vizuálně přitažlivých a interaktivních webových stránek. Využitím možností jazyka HTML5 mohou vývojáři strukturovat obsah sémantickým a přístupným způsobem, zatímco CSS3 jim umožňuje vytvářet vizuální návrhy a responzivní rozvržení. JavaScript naproti tomu dodává webovým aplikacím interaktivitu a dynamickou funkčnost a obohacuje tak uživatelský zážitek.

Front-endové frameworky jako Angular, ReactJS sehrály klíčovou roli při utváření moderního webu. Tyto frameworky nabízejí vývojářům robustní sadu nástrojů a knihoven, které umožňují vytvářet responzivní a škálovatelné webové aplikace. ReactJS se svou komponentovou architekturou usnadňuje efektivní správu a opakovanou použitelnost komponentů uživatelského rozhraní, zatímco Angular poskytuje komplexní platformu pro vytváření rozsáhlých aplikací s funkcemi, jako vstřikování závislostí a obousměrné vázání dat.

Kromě front-endových technologií hrají při poskytování dynamických a datově řízených webových aplikací zásadní roli také back-end technologie. Technologie jako PHP, Python, Ruby, Java, SQL a frameworky jako Laravel, Django, Flask, Ruby on Rails, Spring a NodeJS, umožňují vývojářům zpracovávat data, vytvářet škálovatelné a bezpečné webové aplikace, zpracovávat složitou obchodní logiku a komunikovat s databázemi a externími službami.

Cílem této práce je prostřednictvím zkoumání těchto moderních webových technologií poskytnout základní porozumění jejich významu a dopadu na vývoj webových aplikací. Zkoumáním technologií a jejich výhod si tato práce klade za cíl vybavit čtenáře znalostmi potřebnými pro porozumění vývoje moderního webu.

1 Základy Webových technologií

1.1 Historie a evoluce WWW

WWW (World Wide Web), překlad do českého jazyka je Celosvětová Síť. Místo WWW se také používá termín web. *Web vytvořil roku 1989 anglický informatik Sir Timothy Berners-Lee v době kdy pracoval pro CERN.*[1] (Vlastní překlad). Je to systém vzájemně propojených dokumentů. Tyto dokumenty jsou propojeny skrze hypertextové odkazy a URI za pomocí internetu. Berners-Lee chtěl pro fyziky vytvořit způsob, jak snadněji spolupracovat a sdílet informace. Představoval si systém, který by umožňoval odkazovat na dokumenty a další zdroje přes internet. Pro uskutečnění této představy vyvinul sadu technologií. Tyto technologie jsou HTTP, HTML a první webový prohlížeč, který se jmenoval WorldWideWeb. Web si prošel pár revolučními změnami. Tyto změny byly natolik důležité, že každá z nich začala novou etapu, které se značí jako verze webu. První verze byla vytvořena Berners-Leem a označuje se za Web 1.0, současnou verzí je Web 2.5. Tyto označení ale slouží spíše jako pojmenování pro změnu způsobu využívání webu, než že by byla vytvořena nová verze. Web si prošel zatím třemi verzemi Web 1.0, Web 2.0 a Web 2.5.

1.2 Web 1.0

Jak bylo řečeno Web 1.0, byl vytvořen Sirem Timothy Berners-Leem za účelem usnadnění spolupráce mezi fyziky. Byl statický a sloužit měl pouze pro hledání a zobrazování informací nebo dokumentů pomocí prohlížeče. Web 1.0 byl postaven na třech základních pilířích. Tyto pilíře zahrnovali HTML standard, který definoval formát jednotlivých dokumentů, protokol HTTP definující způsob předávání informací mezi jednotlivými zařízeními a definici URI. Postupem času přibyly další technologie jako XML, XHTML a CSS, nebo také kombinované technologie mezi serverem a klientem jako ASP, PHP, JSP, CGI a PERL. Pro front-end se používal v dnešní době již nepodporovaný Adobe Flash runtime environment nebo Java. Back-end používal JavaScript a VBscript. Web 1.0 pracoval pouze jednostranně, takže pokaždé když na stránce proběhla nějaká změna, byla potřeba stránku obnovit, aby se tato změna zobrazila v prohlížeči uživatele.

1.3 Web 2.0

Změna, kterou s sebou přinesl Web 2.0 je obousměrná komunikace mezi serverem a uživatelem. Obousměrná komunikace umožnila uživatelům přidávat a sdílet vlastní obsah, spolupracovat a komunikovat s ostatními uživateli. Webové stránky nadále není potřeba

obnovovat při každé změně obsahu. S přelomem webu se změnila i sémantika označování webových prvků ze stránek na aplikace, kdy pojem webové stránky vzniknul z původní definice webu jako nástroje sdílení dokumentů, kde každý odkazovaný dokument reprezentoval jednu stránku. S vývojem technologií rostou možnosti a začíná se klást důraz na zážitek uživatele při používání webové stránky, to znamená lepší design a mobilní přístupnost. Důležitou událostí byl vzestup sociálních sítí. Mezi tyto sítě patří především obři jako Facebook, Twitter nebo LinkedIn, kteří se mohou těšit svojí stále rostoucí popularitou. Objevily se také platformy pro sdílení multimédií například všemi známý YouTube a o něco méně známé Vimeo. Pro práci nebo pro sdílení souborů vznikly Google Docs a Dropbox. Tohle vše je uskutečněno pomocí AJAX technologií jako JavaScript, XML, DOM, REST a CSS. V současné době se pohybujeme na přelomu verzí 2.0 a 3.0, dalo by se tedy říci, že současnou verzí je web 2.5.

1.4 Web 3.0 a Sémantický web

Základní ideou, je transformace internetu na jednu globální databázi informací. Informace by měly být ve formátech, které jsou přístupné a srozumitelné jak pro lidi, tak stroje. Toho je dosaženo pomocí nástrojů, jako jsou metadata, ontologie a strojové učení. Když mluvíme o webu 3.0, je také nutno zmínit Sémantický web. Jedná se o označení nového způsobu vyhledávání webů, informací a dokumentů na internetu. Vyhledávání by mělo probíhat podle významu vyhledávaných slov namísto jejich výskytu na webové stránce. Vyhledávače si také budou pamatovat, co uživatel vyhledával v nedávné minulosti a budou tuto informaci přidávat k ostatním, aby nalezly to, co uživatel požadoval. Tohle vše má umožnit bezproblémovou spolupráci mezi službami a aplikacemi. Zatímco Web 2.0 byl o interakci mezi uživateli a uživatelském obsahu, Web 3.0 je o inteligentních datech, přístupu k nim a bezproblémové integrace služeb a aplikací.

2 Základní koncepty vývoje webu

Vývoj webu je proces tvorby webových stránek a aplikací, který se skládá ze dvou částí: front-end vývoj a back-end vývoj. Front-end se specializuje na vizuální prvky webu a back-end se specializuje na prvky, které umožňují jeho chod. Kombinace těchto dvou se poté nazývá full-stack. Kromě těchto konceptů je také třeba zmínit HTTP, sloužící pro přenos dat mezi klientem a serverem a jednu z nejpoužívanějších architektur tedy klient-server architekturu.

2.1 Front-end

Front-end vývoj neboli také client-side v překladu na straně klienta. Má na starosti viditelnou část stránky. Zahrnuje veškeré prvky, které uživatel vidí a se kterými probíhá interakce. Front-end vývojář zastupuje dvě role návrháře a vývojáře. Úkolem návrháře je navrhnout web tak aby vyhovoval zákazníkovi a zároveň byl vhodný pro účely daného webu. Toto může zahrnovat výběr barevné palety, fontu, řádkování, velikost písma, rozmístění prvků na obrazovce a další viditelné prvky. Je třeba brát také ohled na fakt, že v dnešní době obrazovky na kterých je stránka zobrazena nemají stejnou velikost, je tedy vhodné, aby se automaticky přizpůsobovaly velikosti displeje. Například velikost písma na monitoru stolního počítače nemusí vyhovovat pro zobrazení na tabletu nebo telefonu. Druhou rolí je vývojář, tato role vyžaduje znalosti programování, front-end vývojář by se měl tedy orientovat v základních programovacích jazycích pro tvorbu a práci s webovými stránkami. Pro front-end vývoj jsou to jazyky, HTML, CSS a JavaScript. Kromě programovacích jazyků by měl mít znalosti o populárních frameworcích jako React, Bootstrap, Backbone, AngularJS a EmberJS.

2.2 Back-end

Moderní webové stránky a aplikace nemohou fungovat pouze po vzhledové stránce, v dnešní době se od webu očekává více než pouhé zobrazení dokumentu. Každý pořádný web obsahuje některé nebo všechny z těchto funkcí: ověření uživatelů (registrace, přihlášení, odhlášení), panel pro zprávu webu, formuláře, možnost nahrání souboru. Krom těchto funkcí má back-end na starost tvorbu a údržbu serverových částí webové stránky, databáze, skripty na straně serveru, API a logiku aplikace. Back-end pracuje především s daty, jejich přijímáním, zpracováním a ukládáním. Tyto data zah rnují nejen obsah stránky ale i uživatelská data, je tedy potřeba, aby infrastruktura byla dostatečně zabezpečená proti únikům těchto dat. Data je možné ukládat několika způsoby: cookies (malé soubory, ukládané prohlížečem na zařízení uživatele), místní uložiště (prohlížeč dovoluje webovým stránkám ukládat soubory v uživatelském zařízení), relační uložiště (stejné jako místní uložiště ale data zmizí po zavření prohlížeče.), webové uložiště (spojení místního a relačního uložiště.), ukládání do mezi paměti na straně serveru (některé webové frameworky a CMS ukládají často navštěvovaná data do serverové mezi paměti. Díky tomu jsou často používaná data přístupná i bez potřeby připojení k databázi. Je to snaha o lepší výkon webové stránky.). Tyhle metody ukládání dat jsou vhodné pro ukládání malého množství dat, pro větší

množství dat je zapotřebí použití databází. Pro back-end vývoj tedy nestačí umět pouze programovací jazyky pro vytvoření základní logiky, jako jsou PHP, Python, Ruby, Java nebo TypeScript, ale také znalost databází a jazyk zabývající se databázemi jako třeba SQL.

2.3 Full-stack

Kombinací konceptů front-end a back-end vzniká koncept zvaný full-stack. Pro tento způsob vývoje je třeba dostatečná znalost obou součástí vývoje. Tento přístup k vývoji webových stránek a aplikací je populární, přesto je však vhodný pouze u menších projektů. Při práci na rozsáhlých webech je specializace na jednu část výhodou.

2.4 HTTP

V první kapitole byly zmíněny tři základní pilíře a jedním z nich je HTTP (Hypertext Transfer Protocol), do češtiny přeloženo jako protokol přenosu hypertextu. Jak z názvu vyplývá, tak se jedná o protokol, nebo způsob přenosu dat mezi serverem a klientem. Architektura vzniklá jejich propojením se nazývá klient-server architektura nebo klient-server model. *Server a klient spolu komunikují pomocí HTTP zpráv, proto se jim někdy říká HTTP server a HTTP klient. Nejběžnějším klientem je webový prohlížeč*.[8] (Vlastní překlad.) Webové servery uchovávají webové zdroje. Tyto zdroje mohou být statické i dynamické soubory. Statické soubory jsou například textové soubory, obrázky, videa, HTML soubory. Mezi dynamické soubory poté patří například e-shopy, sociální sítě, webové aplikace. Každý soubor má svůj datový typ, a protože těchto typů je příliš mnoho tak je HTTP označkuje, jako MIME typ, kterému se budeme věnovat později v této kapitole. Pro komunikaci mezi sebou klient a server využívají HTTP zpráv. Tyto zprávy se dělí na žádosti a odpovědi. Komunikace začíná žádostí odeslanou klientem, na kterou poté server odpoví odpovědí.

2.4.1 Žádosti

Jsou to zprávy, které požadují po serveru nějakou akci. Tato zpráva je odeslaná klientem a skládá se ze tří částí: startovní řádek, hlavička nebo také záhlaví a volitelný obsah. Startovní řádek obsahuje metodu, cílové URI a použitou verzi HTTP. Verze HTTP je důležitou součástí zprávy, protože se HTTP neustále vyvíjí. Server tedy tuto informaci použije k tomu, aby věděl, jak má požadavek zpracovat a jakou má odeslat odpověď. Hlavička poskytuje dodatečnou informaci o požadavku jako například typ odesílaných dat, uživatelského agenta, který zadává požadavek nebo ověřovací údaje.

- Accept: Říká, jaké typy MIME je klient schopen zpracovat v odpovědi.
- User-Agent: Oznamuje, pomocí kterého klientského softwaru (např. webový prohlížeč) byla žádost odeslána.
- Authorization: Údaje potřebné k zpřístupnění chráněných zdrojů.

Volitelný obsah je využíván metodami *POST* a *PUT* a obsahuje data, která jsou odeslána na server.

Metody jsou příkazy, které komunikují serveru, jakou akci má vykonat. Metody *GET* a *HEAD* se považují za tzv. bezpečné metody, to znamená, že nevykonávají žádnou změnu na serveru.

Vybrané metody:

- *GET* – říká serveru, že klient od něj požaduje reprezentaci daného zdroje
- *HEAD* – je podobná metodě GET, s rozdílem že požaduje pouze záhlaví
- *POST* – odešle serveru data ke zpracování
- *PUT* – uloží obsah žádosti na server
- *DELETE* – odstraní vybraný zdroj
- *OPTIONS* – vrátí HTTP metody, které jsou serverem podporovány pro dané URL
- *CONNECT* – navazuje síťové spojení se serverem
- *TRACE* – stopuje zprávu přes proxy servery k serveru
- *PATCH* – aplikuje částečné změny na zdroj

2.4.2 Odpovědi

Jedná se o typ HTTP zprávy sloužící jako odpověď serveru klientovi. Odpověď se skládá ze třech částí: počáteční řádek, hlavička a obsah neboli tělo. Počáteční řádek obsahuje nejprve HTTP verzi a poté stavový kód. HTTP verze zde zastává téměř identickou roli jako při požadavku. Jedná se o informaci, pomocí které klientské zařízení pozná, jak má s odpovědí naložit, tělo poté obsahuje požadovaný soubor.

Stavové kódy

Stavové kódy jsou složeny ze tří číslic, a slouží k předání informace, zda vše proběhlo v pořádku, nebo jestli a jaká nastala chyba. Mají své kategorie, která je patrná první cifrou daného kódu.

- 1xx – informační zprávy
- 2xx – pozitivní odpověď serveru
- 3xx – přesměrování
- 4xx – chyba na straně klienta
- 5xx – chyba na straně serveru

Zde je příklad vybraných stavových kódů:

- 200 OK: Požadavek byl úspěšný a server odesílá požadovanou informaci.
- 201 Created: Požadavek byl úspěšný a na serveru byl vytvořen nový zdroj.
- 204 No Content: Požadavek byl úspěšný ale nebyl zde obsah k přiložení k odpovědi.
- 400 Bad Request: Server nebyl schopen rozpoznat požadavek, z důvodu nesprávné syntaxe nebo neplatných parametrů.
- 404 Not Found: Požadovaný zdroj nebyl nalezen na serveru.
- 500 Internal Server Error: Server narazil na neočekávaný problém při zpracování požadavku
- 502 Bad Gateway: Server sloužící jako brána nebo zástupce zaznamenal neplatný požadavek od nadřazeného serveru
- 503 Service Unavailable: Server není momentálně schopen vyřídit požadavek z důvodu údržby nebo přetížení.

Hlavička obsahuje dodatečné informace, v případě odpovědi to mohou být:

- Content-Type: Upřesňuje MIME
- Content-Length: Předává informaci o velikosti těla v bytech.
- Server: Identifikuje software použitý serverem pro zpracování požadavku.

2.4.3 HTTPS

„*Za posledních 25 let se Web rozrostl na platformu pro většinu světové komunikace, ať už je to sdílení informací, budování komunit, obchod, vzdělávání, sociální sítě, nebo podpůrné aplikace.*“[52] (Vlastní překlad) Protože se web stal nedílnou součástí životů většiny obyvatelstva, byla potřeba zajistit jeho bezpečnost. Tohle dalo vzniku protokolu HTTPS (Hypertext Transfer Protocol Secure) v překladu bezpečný protokol přenosu hypertextu. Tato verze HTTP protokolu využívá bezpečnostního komunikačního protokolu SSL/TLS (Secure Sockets Layer/Transport Layer Security). Pomocí těchto protokolů HTTPS vytváří zašifrované spojení mezi webovým serverem a webovým prohlížečem. Na rozdíl od HTTP,

který používá pro komunikaci port 80, HTTPS používá pro svou bezpečnou komunikaci port 443. Přechod na HTTPS znamenal ještě jednu změnu, a to v URL kde místo *http://* je třeba použít *https://*.

2.5 URI

URI (Uniform Resource Identifier), český překlad Jednotný Identifikátor Zdroje. Každému internetovému zdroji je přiřazeno URI, jedná se, o adresu nebo lokaci daného zdroje. Pomocí URI jsou tedy klienti s využitím internetového prohlížeče schopni nalézt jakýkoliv zdroj. URI existuje ve dvou variantách. URL (Uniform Resource Locator) a URN (Uniform Resource Name) v překladu Jednotný Lokátor Zdroje a Jednotný Název Zdroje.

2.5.1 URL

Nejběžnější formou URI je URL. Jak z názvu vyplývá, označuje přesnou lokaci daného zdroje. Dalo by se říci, že se jedná o adresu zdroje. Skládá se ze třech částí: Protokolu (jako HTTP nebo HTTPS), jména domény, a cesty k požadovanému zdroji. Příklad: *https://www.priklad.cz/obrazky/priklad.jpg*

2.5.2 URN

URN je používáno pro vyhledávání zdroje podle jména. Jsou navržena tak, aby byly trvalé. Nezáleží tedy na lokaci zdroje nebo zda se jeho lokace změnila, jeho URN zůstane vždy stejné. Na první pohled toto řešení vypadá lépe, nicméně použití URN je pomalejší a je zde také nedostatek standardizace. Absencí jakéhosi standartu vzniká více způsobů jejich zapsání a tím se také zvyšuje obtížnost je implementovat nebo s nimi pracovat. URN se tedy používají převážně pro digitální zdroje jako knihy, články nebo videa, které mohou změnit svou lokaci.

2.6 MIME

Každý zdroj má svůj datový typ, a protože jich je hodně HTTP používá pro jejich označení MIME (Multipurpose Internet Mail Extensions) v překladu do češtiny víceúčelové rozšíření internetové pošty. *Původně byl MIME vytvořen jako řešení problémů, vznikajícími přesouváním zpráv mezi různými elektronickými mail systémy. Protože se velice osvědčil tak jej HTTP začalo používat také. HTTP jej používá k popisu a označení multimediálního obsahu*[8] (Vlastní překlad.) MIME je poznámka pro prohlížeč, která říká, jakým způsobem má s daným objektem nakládat. Tato poznámka může obsahovat například datový typ a jeho

délku. Poznámka o typu se dělí na dvě části, typ nejvyšší úrovně a podtyp. Příkladem může být html soubor, který by byl zapsán jako *text/html*, nebo obrázek jako *image/png*. První slovo je tedy typ nejvyšší úrovně a slovo za lomítkem označuje podtyp. Délka nebo velikost typu je zapsána číselnou hodnotou. Příkladem MIME může tedy být *Content-type: image/png“ „Content-length: 2048*

2.7 Klient-server architektura

Při popisu základních konceptů vývoje webu bylo řečeno, že jednou z nejpoužívanějších architektur je klient-server architektura. Klient-server architektura je počítačový model, ve kterém spolupracují dva nebo více počítačových programů či zařízení. Jedná se o základní koncept v moderní výpočetní technice. Základní ideou je oddělení prezentační vrstvy od vrstvy pro zpracování dat a uložiště. Oddělením vrstev jako aplikační vrstva, databázový přístup, uživatelské rozhraní atd. umožňuje snazší správu a údržbu systému. Každá z těchto vrstev také může být vyvíjena a testována nezávisle na ostatních vrstvách. Změny provedené na jedné vrstvě mohou být provedeny bez ovlivnění jiné vrstvy. Mohou nastat případy, kdy se aplikace stane natolik velkou a složitou, kdy pro lepší výkon bude potřeba rozdělit funkce napříč vrstvami. Proto se klient-server architektura dělí na čtyři úrovně. S každou úrovní přidáváme další vrstvy, což umožňuje z nich soustředit se na určitý úkol. Pokud jedna vrstva zodpovídá pouze za jeden úkol, je snazší spravovat a rozšiřovat systém jako celek. Pro zajištění větší bezpečnosti lze také přidat ochranné vrstvy. Například přidáním vrstvy, která se stará o schvalování klientských požadavků, než je odešle do databáze, je možné zabránit útokům jako SQL injekce. V tomto modelu zastupují účastníci jednu ze dvou rolí. Z názvu této architektury vyplývá, že se jedná o role, klient a server. Klienti jsou zařízení či program, které odesílají požadavky a servery jsou zařízení, které tyto požadavky zpracovávají, a poté odesílají odpověď zpět klientovi. Jedná se o centralizovanou strukturu. To znamená, že veškerá data a služby jsou spravovány centrálním serverem. Při využití tohoto typu se server stává středobodem sítě a veškerá komunikace mezi klienty prochází skrze centrální server. Jeho rolí je spravovat síť a kontrolovat přístup ke zdrojům. Klienti jsou tedy zcela závislí na serveru.

2.7.1 Klient

Klientem je zařízení či aplikace, které odesílají požadavky na server. Klientské zařízení mohou být notebooky, osobní počítače, telefony a tablety nebo jakékoliv zařízení schopné

připojení k internetu. Příkladem nejběžnější klientské aplikace je webový prohlížeč, ale mezi klientské aplikace patří také například MS Word nebo Adobe PhotoShop. Poslední dvě jsou považovány za klientské aplikace, díky jejich propojení s internetem, při jejich využívání je možné kontaktovat online podporu nebo vyhledávat a stahovat různé doplňky jako fonty, štětce a různá nastavení z jim přístupných serverů.

2.7.2 Server

Server je zařízení či program, poskytující služby nebo zdroje klientům skrze síť. Mohou mít formu fyzického stroje, virtuálního stroje nebo cloudové služby. Jejich hlavní funkcí je přijímat a zpracovávat požadavky klientů a poskytnout požadované zdroje či služby. Zatímco klient poskytuje prezentační vrstvu, server pracuje s vrstvou pro zpracování dat a s uložištěm. Je nutno poznamenat že zdroje poskytované serverem nemusí být vždy nutně uloženy na daném serveru. Nejběžnější typy serverů jsou: Webové servery, hostující webové stránky a aplikace, přístupné klientům pomocí internetu. Souborové servery, poskytující centralizované ukládání souborů a přístup ke klientským zařízením, což usnadňuje sdílení a správu souborů. Emailové servery, tyto servery se starají o přenos zpráv mezi klienty. Databázové servery, poskytující přístup k databázím.

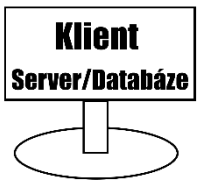
2.7.3 Databáze

Databáze je organizovaná sbírka informací nebo dat, typicky uložených v počítačovém systému. Jsou navrhovány tak aby efektivně ukládaly, načítaly a manipulovaly s velkým množstvím dat a zajistily jejich integritu, bezpečnost a spolehlivost. Existuje více druhů databází a každá je určena pro jiný typ dat nebo způsob jejich ukládání. *Nejpoužívanějším druhem databází jsou relační databáze*[11](Vlastní překlad), které k práci a komunikaci využívají programovací jazyk SQL. Charakteristikou relačních databází je jejich organizace jako soubor tabulek se sloupci a řádky. Databáze jsou obvykle řízeny databázovými systémy, Mezi které patří například MySQL, Oracle Database, mongoDB, Microsoft SQL Server a PostgreSQL.

2.7.4 Tier 1

Nejnižší úrovní klient server architektury je Tier 1 neboli jednoúrovňová architektura, kde celá aplikace běží na jednom zařízení. Uživatel tedy interaguje přímo s aplikací. Práce s tímto modelem je jednoduchá, není zapotřebí jiných zařízení než toho, na kterém model

běží. Je vhodný pro malé aplikace, které nepotřebují rozdělit jednotlivé funkce do vrstev. Protože model běží na jednom zařízení, jedná se o rychlé a efektivní řešení s minimální odezvou a bez potřeby připojení k internetu. Jsou zde ovšem i nevýhody, v případě že aplikace dosáhne velikosti, kdy je potřeba úkoly rozdělit do vrstev, stává se tento model nepraktickým a s jejím vývojem roste obtížnost její správy a údržby. Bezpečnost může být problém u tohoto modelu, protože celá aplikace běží na jednom zařízení, je tedy zranitelná vůči virům nebo malware. Velkým problémem je obtížnost úprav. V případě že je potřeba udělat změny, třeba je udělat v celé aplikaci. Těmto problémům lze zabránit použitím nebo přechodem na vyšší úroveň client-server architektury.

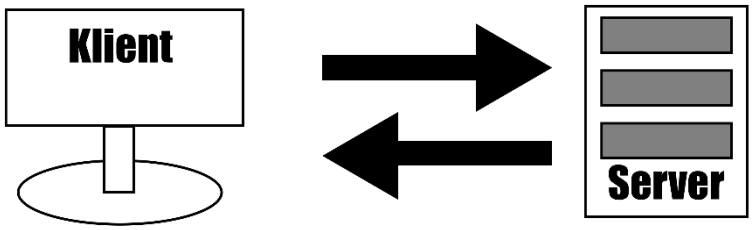


Obrázek 1: Tier 1 (David Grydil)

2.7.5 Tier 2

Tier 2 je model zahrnující dva účastníky, klient a server. Klient je zodpovědný za uživatelské rozhraní a odesílání uživatelských požadavků na server. Server je zodpovědný za ukládání a správu dat, provádění výpočtů a logiky a odpovídání na klientovi požadavky. Rozdělením prezentační a aplikační logiky usnadňuje správu a úpravu aplikace. Vývojáři se tedy mohou soustředit na řešení konkrétních problémů a úkolů což dělá výslednou aplikaci modulární a snadno udržitelnou. Oddělené vrstvy je možné nezávisle rozšiřovat. Díky oddělení aplikační vrstvy je možné přidat bezpečnostní opatření jako třeba firewall nebo ověřovací mechanismy. Každá vrstva může být optimalizována pro výkon, což vede ke zvýšení rychlosti a efektivnosti. Je také možné upravit hardware podle potřeby, například klientský počítač nebude potřebovat tak velké množství výpočetního výkonu a uložiště jako by potřeboval, kdyby bylo použito pouze jedno zařízení. Zatímco server neobsahuje grafické rozhraní a není tedy potřeba grafické karty. Protože tento model pracuje s více zařízeními, jeho sestavení trvá déle. Oddělení prezentační vrstvy a aplikační vrstvy vyžaduje práci na návrhu a konstrukci infrastruktury. Od architektury úrovně dvě nahoru je třeba vytvořit propojení ať už pomocí internetu nebo kabelu. To může způsobit pomalejší chod aplikace, protože data je třeba posílat mezi více zařízeními. Zpomalení může mít i další důvody

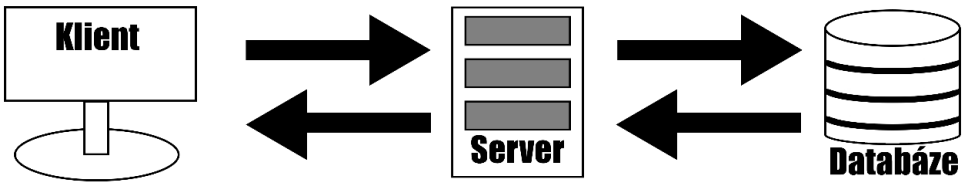
například odezva, limitovaná šířka pásma, data navíc přidána protokolem, výpočet, kódování a dekódování zpráv nebo oprava chyb.



Obrázek 2: Tier 2 (David Grydil)

2.7.6 Tier 3

Tier 3 také známá jako víceúrovňová architektura. Tato architektura se skládá ze třech zařízení/vrstev: klient, aplikační server a databázový server. Server zde hraje roli prostředníka. Veškerá komunikace probíhá skrze něj. Klient komunikuje s aplikačním serverem, v případě potřeby komunikace s databází činí tak skrze prostředníka tedy aplikační server. Aplikační server se stará o výpočetní logiku a zpracování klientských požadavků a zprostředkovává komunikaci mezi částmi modelu. Databázový server se stará o ukládání a správu dat. Výhody jsou stejné jako u dvouúrovňové architektury, škálovatelnost, údržba, flexibilita, bezpečnost. Protože je ale tato architektura větší a komplexnější než předchozí, přináší sebou také řadu problémů. Navrhnout ji správně pro danou potřebu je složitější. S velikostí také přichází větší nároky na finance, je zde totiž potřeba více hardwaru, software, který je vhodný pro takovou architekturu. Stejně jako u dvouúrovňové architektury, může zde nastat zpomalení a výkonu, z důvodu předávání a zpracování dat mezi více body. Tento model je také velice závislí na síťové komunikaci mezi vrstvami, pokud je síť pomalá nebo nespolehlivá může to vážně ovlivnit výkon celého systému.



Obrázek 3: Tier 3 (David Grydil)

2.7.7 N-Tier

Architektura vyšší úrovně než Tier 3 je označována jako N-Tier architektura. Rozdílem mezi Tier 3 a N-Tier je počet úrovní, vrstev nebo zařízení a oproti předchozím modelům zde není

určen maximální počet. V tomto modelu každá vrstva vykonává určitou funkci a vrstvy spolu vzájemně komunikují skrze přesně definované rozhraní. N-tier architektura má lepší škálovatelnost, flexibilitu a snazší údržbu oproti předchozím modelům. Cenou za tyto výhody je ale vetší složitost systému. Bezpečnost zde bývá vyšší než u předchozích modelů, často se totiž přidává více bezpečnostních vrstev a prvků, například umístění datové vrstvy za firewall a omezením přístupu k ní. Monitorování a správa citlivých dat je zde snazší a vývojáři si mohou zvolit nástroje a technologie pro každou úroveň podle potřeby. Jednou z hlavních předností této architektury je spolehlivost, každá úroveň může totiž pracovat nezávisle na ostatních. S dalšími úrovněmi přichází větší cena a komplexnost, výsledkem je tedy více práce při návrhu této architektury. Každá vrstva s sebou přináší další propojení, což se může projevit jako vážný problém, pokud je síť pomalá a nespolehlivá, mohou se vyskytnout potíže s přenosem dat mezi vrstvami a způsobit snížení výkonu. Se správným návrhem je ale možné se tomuto problému vyhnout. Testování správné funkčnosti je časově náročné a může se prodražit. Při testování je totiž potřeba zjistit, zda všechny vrstvy fungují správně, a protože je zde více vrstev zvyšuje se tím tedy čas a peníze. Veškeré výhody a nevýhody této architektury se tedy odvíjí od počtu úrovní, rychlosti připojení a jak dobře byla daná architektura navržena.

3 Front-End Technologie

3.1 HTML

HTML – (Hypertext Markup Language) Hypertextový značkovací jazyk. Hypertextový znamená propojení webových stránek a značek. *Programovací jazyk HTML vytvořil Tim Berners-lee.*[1](Vlastní překlad) První verze se jmenovala HTML1.0 a v současné době se používá verze HTML5.0. HTML je páteř a základní kámen vývoje každé webové stránky. Slouží k navrhnutí základního rozložení stránky. Zápis programu je ve formě značek, které se zapisují do úhlových závorek a dělí se na párové značky (sebe uzavírající) a nepárové značky (prázdné). Některé značky mají také své atributy, jedná se o další vlastnosti, které lze elementu přidělit. Prohlížeč značky nevykresluje, ale využívá je jako instrukce k vykreslování obsahu. Každý HTML dokument začíná prohlášením *<!DOCTYPE html>*, které webovému prohlížeči říká, že se jedná o HTML dokument. Pomocí HTML může vývojář do jisté míry stylizovat vzhled stránky, nicméně jeho možnosti jsou značně omezené, a protože *HTML nikdy nebylo určeno k formátování*[15](Vlastní překlad.) tak se tato část přenechává CSS, který je na to uzpůsobený.

Párové značky jsou vždy zapsány v úhlových závorkách a je třeba je ukončit. Jejich syntax je následující <*značka*> Obsah </*značka*>. Příklady nejpoužívanějších párových značek:

- <*html*> </*html*> - začátek a konec HTML dokumentu
- <*head*> </*head*> - hlavičková část webové stránky
- <*body*> </*body*> - tělo webové stránky
- <*h1-6*> </*h1-6*> - slouží k vytvoření nadpisu úrovně 1,2,3,4,5 nebo 6.
- <*p*> </*p*> - paragraf textu
- <*a*> </*a*> - hypertextový odkaz, který může vést na jinou webovou stránku nebo určitou část stránky nebo i emailovou adresu.
- <*div*> </*div*> - vytvoří „pouzdro“, které slouží k seskupení HTML elementů a přidá styly celému tomuto seskupení

Nepárové značky jsou zapsány v úhlových závorkách. Říká se jim nepárové, protože nemají ukončovací značku, jejich syntax je tedy pouze <*značka obsah*>. Příklady nejvyužívanějších nepárových značek:

- <*img*> - vkládá obrázek na webovou stránku
- <*br*> - přidává řádkovou mezeru
- <*input*> - vytváří pole pro vstup
- <*link*> - odkaz k vnějšímu zdroji
- <*meta*> - předává metadata o HTML dokumentu jako klíčová slova a popisek pro internetové vyhledávače

Atributy poskytují další informace o elementu jako je zdroj pro obrázek, cíl pro odkaz nebo styl daného elementu. Zápis atributu probíhá ještě před ukončením první úhlové závorky značky. Příklady nejvyužívanějších atributů:

- *class* – definuje, do kterých tříd element patří
- *id* – unikátní identifikátor pro element
- *src* – specifikuje zdroj URL pro obrázek, zvuk nebo video soubor.
- *href* – specifikuje URL na které odkaz odkazuje
- *alt* – přidává popis k obrázku pro asistenční technologie
- *title* – přidává popis elementu při najetí kurzorem na daný element
- *style* – přidává elementu vložené styly
- *target* – určuje, jestli otevřít odkaz v novém okně nebo tabu

3.1.1 Ukázka HTML kódu

Na ukázce níže lze vidět začátek HTML dokumentu. Sekce <*head*> obsahuje metadata pro stránku. Metadata jsou údaje o HTML dokumentu, které se nezobrazují. V tomto případě se jedná o znakovou sadu *UTF-8*, titulek stránky, propojení s CSS dokumenty *styles.css* a *lightbox.css* a font.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Předina</title>
  <link rel="stylesheet" href="styles.css" />
  <!-- Font -->
  <link rel="preconnect" href="https://fonts.googleapis.com" />
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
  <link href="https://fonts.googleapis.com/css2?family=Kumbh+Sans:wght@400;700&display=swap" rel="stylesheet" />
  <link rel="stylesheet" href="lightboxx.css" />
</head>
```

Obrázek 4: HTML kód – hlavička (David Grydil)

Kód pokračuje sekcí <*body*>, ve které je zapsán veškerý obsah stránky. Pro lepší orientaci a stylizování jsou jednotlivé části odděleny a uzavřeny pomocí HTML značky <*div*>, kterým je přiřazena třída atributem <*class*>. Pomocí kaskádových stylů je poté možné tyto části upravovat zavoláním názvu třídy. V příkladu jsou také využity značky <*a*>, pro vytvoření odkazu, <*li*>, pro vytvoření seznamu a <*img*>, pro vložení obrázku.

```
<body data-bs-spy="scroll" data-target=".navbar">
  <header>
    <div class="navigace-obal">
      <div class="navigace-logo">
        <a href="#" class="logo">Předina</a>
      </div>
      <div class="bars">
        <div class="bar"></div>
        <div class="bar"></div>
        <div class="bar"></div>
      </div>
      <ul class="navigace">
        <li class="navigace-link">
          <a href="https://www.brodek.cz" target="blank">Brodek u Prostějova</a></li>
        <li class="navigace-link">
          <a href="https://www.dobrochov.cz" target="blank">Dobrochov</a></li>
        <li class="navigace-link">
          <a href="https://www.hradcany-koberice.cz" target="blank">Hradčany-Kobeřice</a></li>
        <li class="navigace-link">
          <a href="https://www.ondratice.cz" target="blank">Ondratice</a></li>
        <li class="navigace-link">
          <a href="https://www.otaslavice.cz" target="blank">Otaslavice</a></li>
        <li class="navigace-link">
          <a href="https://www.vincencov.cz" target="blank">Vincencov</a></li>
        <li class="navigace-link">
          <a href="https://www.vranovicekelcice.cz" target="blank">Vranovice-Kelčice</a></li>
        <li class="navigace-link">
          <a href="https://www.vresovice.cz" target="blank">Vřesovice</a></li>
      </ul>
    </div>
  </header>
```

Obrázek 5: HTML kód – tělo (David Grydil)

```
<!-- Navbar -->
<div class="container">
  <nav class="navbar">
    <div class="navbar-obal">
      <div class="ukazatele">
        <ul>
          <li><a href="#uvod">Úvod</a></li>
          <li><a href="#zajimavosti">Zajímavosti</a></li>
          <li><a href="#fotogalerie">Fotogalerie</a></li>
          <li><a href="#projekty">Projekty</a></li>
          <li><a href="#komentare">Komentáře</a></li>
        </ul>
      </div>
      <div class="propagace">
        <a href="/test4/obrazky/letak.jpg" data-lightbox="letak" data-title="letak">
          <img src="/test4/obrazky/letak.jpg" /></a>
        <a href="/test4/obrazky/letak2.jpg" data-lightbox="letak" data-title="letak2">
          <img src="/test4/obrazky/letak2.jpg" /></a>
      </div>
      <div class="kontakty">
        <h3>Kontakty</h3>
        <div class="kontakty-predseda">
          <h4>Předseda DSO Mikroregion Předina</h4>
          <h5>Mgr. Bohuslav Koštanský</h5>
          <p>Tel: +420 734 319 001<br />e-mail: starosta@ondratice.cz</p>
        </div>
        <div class="kontakty-mistopredseda">
          <h4>Místopředseda DSO Mikroregion Předina</h4>
          <h5>Marek Hýbl</h5>
          <p>Tel: +420 725 131 146<br />e-mail: starosta@otaslavice.cz</p>
        </div>
      </div>
    </div>
  </nav>
</div>
```

Obrázek 6: HTML kód – navbar (David Grydil)

Čistý HTML dokument bez CSS stylů vypadá takto. Samozřejmě by do jisté míry bylo možné upravit vzhled stránky, nicméně jak bylo řečeno HTML k tomu nebylo určené a jeho možnosti jsou tedy omezené nemluvě o nepraktičnosti.

Obrázek 7: Ukázka HTML (David Grydil)

3.2 CSS

CSS – (Cascading Style Sheets) v překladu kaskádové styly. *Kaskádové styly, jsou jazyk, který popisuje prezentaci dokumentů napsaných ve značkovacích jazycích jako HTML nebo*

XML.[15] (Vlastní překlad.) Tento jazyk vytvořil Håkon Wium Lie, jeho první verze byla CSS a v dnešní době používáme verzi CSS3. Základy CSS jsou poměrně jednoduché, většina názvů elementů totiž vychází z anglického jazyka. Kaskádové styly přidávají k elementům informace nebo pravidla o jejich formátování, nahrazují tedy atributy z jazyka HTML. CSS rozšiřuje možnosti stylizace a s větší volností dovoluje manipulovat se vzhledem stránky. Umožňují formátování od jednotlivých částí jako například nadpisů, paragrafů, obrázků až po nastavení formátu a stylu celé webové stránky nebo dokonce i několika webových stránek najednou. CSS je možné zapisovat přímo do HTML souboru, nicméně jeho hlavní výhodou je právě možnost opakovaného použití napříč celým webem, pro využití této výhody je potřeba vytvořit oddělený CSS soubor a provázat jej s HTML.

3.2.1 Tvorba CSS

Tvorba CSS probíhá pomocí tabulky stylů, která je oddělena od vlastního obsahu stránky. Proto, aby tabulka byla funkční, je třeba použít selektor. Selektor je informace, která nám říká, kde se má daný styl použít. Pro použití musí styl obsahovat selektor a pravidla zapsaná ve složených závorkách. Tabulka kaskádových stylů je tvořena třemi způsoby, vnitřní, vnější a vložené.

Vnitřní styly

Styly se zapisují do párového elementu <*style*>, který se nachází v hlavičce. Do této části se píše jeden nebo více párových elementů <*style*> a každý z nich může obsahovat definici jednoho nebo více stylů. Nevýhoda vnořených stylů je, že tyto definice a pravidla se vztahují pouze na jednu webovou stránku.

Vnější styly

Pokud jsou styly zapsány do samostatného CSS souboru s příponou *.css*. Protože je to soubor vytvořený pouze pro styly CSS nepoužívá se zde párový element <*style*> a obsahuje pouze samotné definice stylů. Soubor CSS připojíme k webové stránce pomocí nepárového elementu <*link*>. Tento element má dva atributy *rel a href*.

- *rel* udává typ stylu, pro CSS se používá *"stylesheet"*.
- *href* je odkaz na CSS soubor.
- Zápis tedy vypadá nějak takto: *<link rel="stylesheet" href="styly/css_styly.css">*
- Příklad kódu pro úpravu nadpisů úrovně *h1*:

Vložené styly

Styly jsou zapsány v HTML souboru a jsou zapsány přímo v elementu, který upravují. Toto se dělá přidáním *style* atributu k danému elementu. Atribut *style* může obsahovat jakoukoliv CSS vlastnost. Tenhle způsob použití CSS je sice jednoduchý, ale ztrácí se tím výhody, kvůli kterým CSS vzniklo a používá se.

3.2.2 Důvody pro využívání CSS

Jedním z největších problémů HTML byla malá možnost stylizace a potřeba kód opakovat u všech elementů na každé webové stránce. Zapsáním stylů do externího souboru vzniká možnost využití již napsaného kódu bez potřeby jeho opakování. To vytváří několik výhod, přehlednější a kratší kód, díky absenci zmíněného opakování. Jednotný styl napříč celým webem. Při úpravě webu stačí upravit pouze jeden soubor a změna se promítne na všechny stránky využívající daný soubor. To umožní vývojáři ušetřit spoustu času. Kaskádové styly nešetří čas pouze vývojáři ale také výpočetní čas počítače, protože styl značky stačí popsat jednou a poté pouze aplikovat při každém jejím použití. Kaskádové styly také obsahují více atributů než HTML, to znamená, že vývojář má více možností při designování webové stránky.

3.2.3 Ukázka CSS kódu

V ukázce HTML kódu byla ukázka, jak vypadá webová stránka bez CSS stylů. Obrázek 8 ukazuje úvodní část CSS kódu. V této části je využita jedna z výhod využití pro stylizaci CSS místo HTML. Jedná se o možnost určení stylů jednotlivých značek napříč celým dokumentem. Kromě selektorů odkazující se na značky použité v HTML dokumentu je zde využit selektor *, tento selektor říká, že styly, které obsahuje, platí pro celý dokument, pokud nebude určeno jinak. V příkladu je také vidět použití funkce *url*, v tomto případě je pomocí něj vložen obrázek na pozadí elementu *h1*.

```css
*{
    margin: 0;
    font-family: 'Kumbh Sans', sans-serif;
}
body{
    height: 100%;
    display: grid;
    background-color: #9a7b4f;
}
a{
    text-decoration: none;
    color: #fff;
}
p{
    max-width: 100ch;
}
h1{
    text-align: center;
    background-color: #2e1503;
    background-image: url("./obrazky/wood-pattern.png");
    border: 2px solid #2e1503;
    padding: 5px;
    color: #fff;
}
li{
    list-style-type: none;
}
img{
    display: block;
    max-width: 100%;
    border: 5px solid #2e1503;
}
```

Obrázek 8: CSS kód – celý dokument (David Grydil)

Kromě HTML značek je také možné pomocí selektorů stylizovat jednotlivé třídy. Jak ukazuje Obrázek 9. Tato konkrétní část kódu se věnuje stylizaci navigační lišty. Kromě standartní stylizace jako úpravy barev, velikostí a pozice, je zde také přidán jakýsi prvek responzivity. Selektor *:hover* přidává možnost změny stylu po najetí myší na daný danou značku nebo třídu. V tomto případě je to snížení jasu značky *a* ve třídě *navbar-obal* na 70% při najetí kurzoru. Obrázek 10 ukazuje, jak může stránka vypadat při přidání CSS stylů.

```css
/* Navbar */
.container{
    position:fixed;
    left:0;
    top:72px;
    bottom:0;
    width:200px;
    height:100%;
    padding: 1rem;
    background-color: #2e1503;
    background-image: url("./obrazky/wood-pattern.png");
    color: #fff;
}
.ukazatele ul{
    font-size: 1.5rem;
    display: grid;
    gap: 0.5rem;
}
.ukazatele li{
    margin-left: -30px;
    width: fit-content;
    background: #2e1503;
}
.navbar-obal{
    display: grid;
    gap: 2rem;
}
.navbar-obal a:hover{
    filter: brightness(70%);
    -webkit-filter: brightness(70%);
}
.propagace{
    display: grid;
    gap: 1rem;
}
.propagace img{
    border: 5px solid #2e1503;
}
.kontakty{
    border: 5px solid #2e1503;
}
```

Obrázek 9: CSS kód – třídy (David Grydil)

Obrázek 10: Ukázka CSS (David Grydil)

3.3 JavaScript

JavaScript patří mezi světově nejpopulárnější programovací jazyky.[17] (Vlastní překlad). Jedná se o dynamický, jedno vláknový, skriptovací jazyk. To znamená, že není předkompilovaný, data a proměnné se mohou změnit kdykoliv a proměnné není třeba

definovat. Jeho vyhodnocení probíhá řádek po řádku v reálném čase. Pro práci a běh JavaScript kódu je zapotřebí prostředí, buď programovací prostředí, ve kterém daný kód píšeme nebo v dnešní době mají prohlížeče vlastní engine, například V8 JavaScript engine u Google Chrome, který toto vyhodnocení umožňuje. JavaScript se používá pro větší dynamičnost stránek. Dovoluje nám implementovat další funkce stránky za pomocí vlastních skriptů nebo API. Stránky využívající JavaScriptu jsou více interaktivní a řízené událostmi. JavaScript lze do HTML souboru přidat několika způsoby například vnitřně, zvenčí anebo jako hodnotu některého z atributů HTML elementu. JavaScript se využívá jak při front-end vývoji tak při back-end vývoji. Jako front-end poskytuje objekty pro ovládání prohlížeče a jeho DOM. Třeba přidáním možnosti aplikaci umístit prvky do HTML formuláře a reagovat na události způsobené uživatelem, jako je kliknutí myší, vyplnění formuláře nebo čistá navigace na stránce. JavaScript se často používá v kombinaci se svými frameworky a knihovnami. Nejpoužívanější frameworky a knihovny při front-end vývoji webové stránky jsou AngularJS, ReactJS, VueJS a jQuery. Jak bylo zmíněno JavaScript nalezne své uplatnění i v back-end vývoji, například v kombinaci s prostředím Node.js nebo opět s knihovnou jQuery.

Vnější JavaScript

Stejně jako u CSS tento způsob přidání JavaScript kódu vyžaduje připojení vnějšího souboru s příponou *.js* obsahující JavaScript kód. Scripty z JavaScript souboru připojíme napsáním názvu souboru do HTML elementu *src*. Tento způsob usnadňuje údržbu a vytváří lepší přehlednost kódu, protože HTML a JavaScript soubory jsou oddělené. JavaScript soubory uložené v mezi paměti urychlují načítání webové stránky. Stejně jako u CSS, využití externího souboru nám dovoluje použít kód vícekrát bez nutnosti jej psát stále dokola pomocí připojení k více HTML souborům.

Vnitřní JavaScript

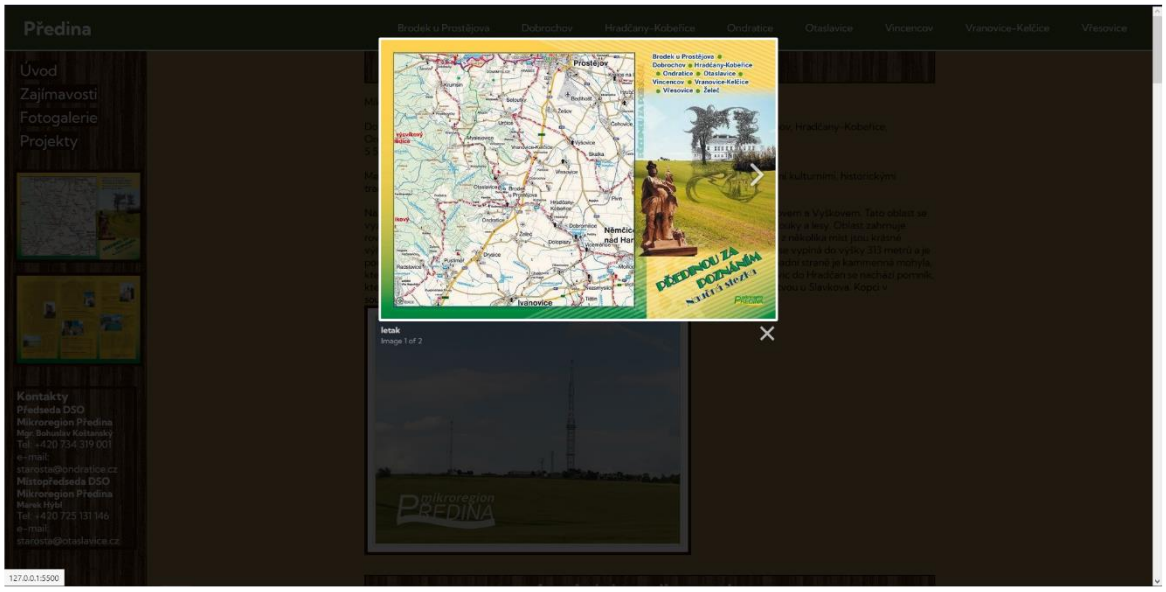
JavaScript je zapsán přímo do HTML souboru pomocí značky <script> a může být zapsán v hlavičce i tělu.

3.3.1 Ukázka využití JavaScriptu

Přidáním JavaScriptu je možné přidat více interaktivnosti. Obrázek 11 a Obrázek 12 ukazují přiblížení obrázku při kliknutí na něj. Tohoto efektu je dosaženo pomocí LIGHTBOX. LIGHTBOX je kombinace CSS a JavaScript kódu, kterou vytvořil autor Lokesh Dhakar.

Kód je příliš dlouhý pro jeho ukázku v této práci proto bude ukázka JavaScript kódu viděna až v kapitole Pokročilá témata webových technologií.

Obrázek 11: JS – před kliknutím (David Grydil)



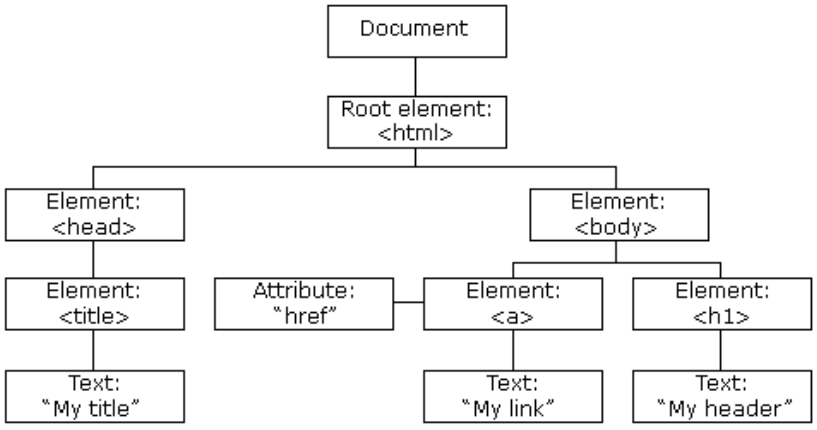
Obrázek 12: JS – po kliknutí (David Grydil)

3.4 DOM

DOM (Document Object Model) je objektová reprezentace XML a HTML dokumentu. Kvůli nekompatibilitě HTML a JavaScriptu. W3C (World Wide Web Consortium) se rozhodlo pro standardizaci, a tak vznikl W3C DOM. Protože JavaScript nerozumí čistému HTML dokumentu, DOM se používá jako reprezentant HTML dokumentu ve formátu, kterému JavaScript rozumí. DOM reprezentuje stejný soubor jako HTML dokument

s rozdílem, že HTML značky jsou zde reprezentovány jako objekty. DOM má formu připomínající strom. Každá větev stromu končí uzlem a každý uzel obsahuje objekty. DOM se skládá z:

- Window objekt – objekt prohlížeče a je vždy na vrcholu hierarchie. Je automaticky vytvořen prohlížečem. Nastavuje a zpřístupňuje veškeré vlastnosti a metody prohlížeče. Vlastnost Window objektu je Document objekt.
- Document objekt – HTML dokument se stává Document objektem ve chvilce kdy je načten do Window objektu. Document objekt má různé vlastnosti odkazující na objekty, které povolují přístup a modifikaci k obsahu webové stránky. Kdykoliv je potřeba přístup k nějakému elementu na HTML stránce, vždy se musí začít zpřístupněním Document objektu.
- Form objekt – je reprezentován *form* značkami.
- Link objekt – je reprezentován reprezentuje *link* značkami.
- Anchor objekt – je reprezentován *href* značkami.



Obrázek 13: Model HTML DOM (W3schools)[20]

4 Populární front-end frameworky

4.1 Angular

Angular je open-source, front-end framework založený na programovacím jazyku TypeScript. Angular je nástupcem frameworku AngularJS. Oproti svému předchůdci, který byl založen na MVC (Model-View-Controller) architektuře, Angular používá architekturu založenou na komponentech. Komponenty jsou stavební bloky pro aplikaci. S pomocí CSS selektorů je možné definovat, jak tyto komponenty použít v šabloně. Klíčovou vlastností je využití jednosměrného toku dat, kdy data v aplikaci proudí pouze jedním směrem. Tím je

myšleno, že dítě nemůže přímo ovlivňovat data svého rodiče, zatímco rodič může předávat data svému dítěti. To zajišťuje konzistentní a spravovatelný tok dat, který je jasný a předvídatelný. Podobné techniky využívá framewok Vue.js, kde dítě vytváří události, skrze které komunikuje s komponenty rodiče.

4.2 ReactJS

ReactJS je open-source knihovna programovacího jazyku JavaScript, vyvinuta společností Facebook. Je převážně určena k vývoji uživatelského rozhraní. ReactJS je znám pro svůj deklarativní styl programování. Deklarativní styl programování se soustředí na popis, čeho by měl kód dosáhnout. V případě ReactJS tedy jak by mělo dané uživatelské rozhraní vypadat. Toho lze dosáhnout použitím JSX rozšíření, které umožňuje zápis a kombinování HTML kódu ve stejném dokumentu zároveň s JavaScriptem. Opakem deklarativního stylu by byl imperativní styl. Kód napsaný v imperativním stylu říká počítači krok za krokem instrukce k dosažení požadovaného výsledku. Deklarativní kódy bývají obecně přehlednější a snadněji se ladí. ReactJS je založený na komponentech. To znamená, že při tvorbě uživatelského rozhraní se rozhraní rozloží na malé části. Tyto komponenty se poté programují zvlášť, a nakonec se poskládají dohromady, aby společně utvářeli jeden celek, kterým je uživatelské rozhraní. Protože se programují samostatně, je možné je také samostatně upravovat, takže není třeba předělávat kód pro celé uživatelské rozhraní. Asi největší výhodou ReactJS je využití virtuálního DOM. Virtuální DOM je odlehčená verze DOM dokumentu, reprezentující reálný DOM v paměti. Protože je rychlejší změnit virtuální DOM než reálný, změny v aplikaci se provedou nejprve ve virtuálním. Po provedení změn ReactJS porovná nový virtuální DOM s předchozím virtuálním DOM, aby našel minimální počet změn, potřebných úpravě reálného DOM. Jakmile jsou tyto změny identifikovány, ReactJS provede změny v reálném DOM, což má za výsledek novou, upravenou verzi uživatelského rozhraní. Využití virtuálního DOM tedy sebou přináší dvě výhody. Minimalizuje se potřebné množství aktualizací reálného DOM dokumentu což má za výsledek rychlejší a bezproblémové využívání webu. Vývojář se nemusí starat o změnu DOM dokumentu ručně, protože ReactJS se o tyto změny postará sám. ReactJS je JavaScript knihovna využívající JSX rozšíření, takže pro plné využití této knihovny jsou znalosti HTML,CSS a JavaScript takřka povinností.

4.3 VueJS

VueJS je open-source framework programovacího jazyku JavaScript pro tvorbu reaktivního uživatelského prostředí, v mnoha ohledech je podobný knihovně ReactJS. VueJS je progresivní framework, tím je myšleno, že jej lze postupně implementovat do již existujících projektů, bez potřeby přepisovat celý kód od začátku. Svou popularitu si získal nejen svojí flexibilitou a efektivním přístupem při vývoji uživatelských rozhraní ale hlavně svojí jednoduchostí. Stejně jako ReactJS, VueJS využívá deklarativního stylu programování a architekturu založenou na komponentech neboli dělení webu či aplikace na malé části zvané komponenty, programovat je samostatně a poté spojit do jednoho většího a složitějšího celku. Hlavním principem VueJS je jeho reaktivní systém. Změny provedené v základních datech se automaticky promítnou do uživatelského rozhraní, zmizí tím potřeba ručně upravovat DOM. Tohoto je dosaženo využitím mechanismu reaktivní vazby dat, která zajišťuje synchronizaci mezi daty a uživatelským rozhraním. VueJS také nabízí vývojáři šablonu, která jednoduše kombinuje HTML a JavaScript. Za použití směrnic *v-bind* a *v-if*, může vývojář propojit data s uživatelským rozhraním a podmínečně vykreslovat prvky. Směrnice jsou speciální atributy, které umožňují přidat dynamické chování HTML elementům. Kromě tohoto, směrnice přináší elementům další funkce jako zpracování událostí a dynamické vykreslování. VueJS s sebou přináší vlastní CLI (command-line-tool) v češtině nástroj příkazového řádku s názvem Vue CLI.

5 Back-end Technologie

5.1 PHP

PHP je akronym pro Hypertext Preprocessor. Je to open-source, server-side, skriptovací jazyk, který se používá pro vývoj webových stránek. V současné době je nejnovější verze PHP 8. Kód PHP je na rozdíl od HTML vyhodnocen na straně serveru a do prohlížeče se odešlou pouze výsledky. Podporuje hlavní protokoly jako HTTP Basic, HTTP Digest, IMAP, FTP a další. PHP kód je možné zapisovat přímo do HTML souborů pomocí značky *<?php* obsah *?>* Kromě HTML, PHP soubory podporují i další client-side (front-end) skriptovací jazyky jako CSS nebo JavaScript. PHP se může pyšnit vlastnostmi jako rychlost, efektivita, bezpečnost, flexibilita, silná podpora knihoven a propojitelnost s databázemi. PHP je dostatečně silný, aby udržel WordPress nebo jiné CMS. Ukázka PHP kódu je vidět v kapitole Pokročilá témata webových technologií.

5.2 Python

Python je open-source, multi-paradigmatický, dynamický skriptovací jazyk vysoké úrovně. Multi-paradigmatický znamená, že podporuje více než jedno programovací paradigma a vysoká úroveň znamená, že je více abstraktní a přátelský k uživateli oproti jazyku nízké úrovně, který je více strojový. Python byl vytvořen tak aby byl čitelný, snadno se do něj zapisovalo a poskytovalo vývojáři co nejvíce volnosti. Tato volnost je umožněna díky množství paradigmat, která Python podporuje. Mezi podporovaná paradigmata patří: procedurální paradigma, objektově orientované paradigma, funkcionální paradigma, imperativní paradigma, deklarativní paradigma a aspektově orientované paradigma. Python je vhodný pro zpracovávání intenzivních operací ale jeho rychlost je menší než například rychlost Node.js. Přináší sebou velkou sadu předem vytvořených modulů a funkcí, které vývojář může využít při práci na běžných úkolech, jako čtení a zápis do souborů, vytváření sítí nebo vývoj webu či webové aplikace. Python je multi-platformní, je tedy možné jej spustit na více operačních systémech jako Windows, macOS a Linux. Je třeba ale poznamenat že ne vždy kód psaný pro jednu platformu bude fungovat beze změny na platformě druhé, malé změny jsou tedy ve spoustě případů potřeba. Python obsahuje rozsáhlou základní knihovnu, kromě této knihovny se může Python pyšnit také bohatou podporou knihoven třetí strany, je tedy možné jej využít ve spoustě oblastí informačních technologií například: ve vědeckých a číselných výpočtech, vývoj webu, tvorba desktopového grafického uživatelského rozhraní, byznys aplikace a softwarový vývoj. Světové weby jako Google, YouTube, Instagram, Dropbox, Pinterest, Spotify a Reddit využívají Python pro svoji back-end infrastrukturu. I přes všestrannost tohoto jazyka a jeho rozsáhlou základní knihovnu je však doporučeno využívat jeho frameworků a knihoven, kterých je díky jeho popularitě dostatek. Pro vývoj webu jsou k dispozici třeba frameworky Django a Flask.

5.3 Ruby

Ruby je open-source, dynamický, multi-paradigmatický skriptovací jazyk, Řídí se motem „Každý problém má více řešení.“. Objektově orientované paradigma je zde vzato do extrému, vše je v Ruby vnímáno jako objekt, každá část informace či kódu může mít vlastní vlastnosti a akce. Jedná se o velice flexibilní jazyk, který vývojáři dovoluje přidávat, odstraňovat nebo upravovat jeho základní části. V Ruby není třeba ručně přidělovat a odebírat paměť, k automatickému řízení paměti používá tzv. sbírání odpadků. Sbírání

odpadků (anglicky garbage collection) je způsob znovu získání paměti, která již není programem používána. Ruby dovoluje tzv metaprogramování to znamená, že je možné napsat kód, který bude upravovat sám sebe nebo jiné části programu během jeho běhu. Třídy v ruby dědí pouze z jedné třídy výše v hierarchii (Rodiče) což je podstatný rozdíl oproti jiným objektově orientovaným jazykům. Stejně jako Python, Ruby se může pyšnit rozsáhlou základní knihovnou. Jeho nejznámější knihovnou je RubyGems a pro vývoj webu se nejčastěji používá framework Ruby on Rails. Ruby je využíváno weby jako GitHub, Airbnb a Twitch.

5.4 Java

Java je objektově orientovaný staticky psaný jazyk. Staticky psaný znamená, že je třeba určit typy proměnných, na rozdíl od Pythonu a Ruby, proměnné jsou zde deklarovány a kontrolovány při kompilování. Java se dělí na čtyři edice: Java SE (Standart Edition), JavaEE (Enterprise Edition), Java ME (Micro Edition) a Java Card. Java SE je rozdělena na dvě části, a to OpenJDK a OracleJDK. OpenJDK je open-source verze JavaSE a obsahuje JDK (Java Development Kit v češtině Java vývojářská souprava), a JRE (Java Runtime Environment v češtině běhové prostředí Java). OracleJDK je komerční verzí JavaSE, která obsahuje další funkce a nástroje, které nejsou přítomny v OpenJDK. Java SE je základní edice, která vytváří základ pro ostatní edice. Poskytuje základní API a běhové prostředí potřebné pro vývoj. Java EE také známá jako Jakarta EE je rozšíření určené přímo pro velké korporátní aplikace. Java ME je zjednodušenou verzí Javy, navržena pro slabší zařízení jako mobilní telefony. Java Card je edice určena pro chytré karty a bezpečnostní prvky například SIM karty, kreditní karty, nebo přístupové karty. Mezi populární vlastnosti tohoto jazyku patří nezávislost na platformě. Kompilace probíhá v bytovém kódu a díky tomu může Java běžet na jakémkoliv zařízení. Java podporuje multi-threading (v češtině více vláken), které dovoluje aplikaci provádět více úkolů zároveň. Stejně jako Ruby, Java využívá sbírání odpadků. Java poskytuje obsáhlou sadu knihoven známou jako JSL (Java Standard Library) v češtině standartní Java knihovna, která obsahuje třídy a metody pro běžné programovací úkoly. Java je využívána společnostmi jako Amazon, Google, LinkedIn, Netflix, Twitter, eBay, PayPal, Spotify.

5.5 SQL

SQL (Structured Query Language) v češtině Strukturovaný Dotazovací Jazyk. Je to databázový jazyk, který byl v letech 1986 a 1987 standardizován organizací ISO. Slouží výhradně pro komunikaci a práci s relačními databázemi. SQL patří mezi neprocedurální jazyky to znamená, že vývojář zadává, jaké informace požaduje místo způsobu, jak je získat. S daty a databázemi se v SQL manipuluje pomocí příkazů, tyto příkazy vychází z anglických slov a dělí se do čtyř základních kategorií: DML, DDL, DCL a TCL. Populární databázové systémy využívající SQL jsou MySQL, Oracle Database, mongoDB, Microsoft SQL Server a PostgreSQL. Nutno podotknout že každý z nich používá lehce jiný dialekt.

- DML (Data Manipulation Language) – Příkazy sloužící k manipulaci s daty
    - *SELECT* – výběr dat z databáze
    - *INSERT* – vložení dat do databáze
    - *UPDATE* – editace dat v databázi
    - *DELETE* – odstraňuje data nebo řádky tabulek z databáze
    - *CALL* – volá uložené procedury
    - *LOCK TABLE* – zamyká celé tabulky, zámek pro čtení (*READ*, zámek pro zápis (*WRITE*)
- DDL (Data Definition Language) – Příkazy pro definici databázových struktur
    - *CREATE* – vytváří objekty databáze (i databázi samotnou)
    - *ALTER* – mění strukturu již vytvořených databázových objektů
    - *DROP* – odstraní objekty databáze (tabulky, indexy, databáze)
    - *TRUNCATE* – odstraní veškerá data z tabulky databáze
    - *COMMENT* – přidává komentáře k objektům databáze
    - *RENAME* – mění název tabulky
- DCL (Data Control Language) – Příkazy pro správu uživatelských práv
    - *GRANT* – nastavuje uživatelská práva k databázi
    - *REVOKE* – odebírá přístupová práva přidělená pomocí příkazu GRANT
- TCL (Transaction Control Language) – Příkazy pro řízení transakcí.
    - *COMMIT* – potvrzení provedení transakce
    - *SAVEPOINT* – nastaví záchytný bod transakce, ke kterému se lze během transakce vrátit

- *ROLLBACK* – zruší transakci a vrátí se zpět do původního stavu nebo k záchytnému bodu

5.6 Node.js

Pro spouštění JavaScript kódu mimo webový prohlížeč je třeba běhového prostředí Node.js. Node.js tedy umožňuje spustit JavaScript kód na serveru. Díky tomu je možné napsat full-stack aplikaci pouze v JavaScriptu bez potřeby jiného back-end jazyku. Vývojář pracující s Node.js má možnost využít NPM (Node Package Manager), pro implementací modulů třetí strany, pokud je požadovaný modul registrován v NPM, je snadné jej do projektu přidat. Přes podobnost s knihovnou, NPM spíše zastupuje roli správce knihoven, balíčků a modulů. Veškeré API Node.js jsou asynchronní což znamená že server využívající Node.js nikdy nečeká na data z API. Přestože se nejedná o programovací jazyk i Node.js má svůj vlastní framework Express.js. Jedná se o framework, který adaptuje minimalistický přístup. Obsahuje základní sadu funkcí webové aplikace. Při vývoji má vývojář volnost při volbě nástrojů a větší kontrolu nad návrhem a architekturou aplikace. Node.js je využíván například společnostmi Netflix, LikedIn, PayPal nebo třeba i NASA (pro některé projekty).

6 Populární back-end frameworky

6.1 Laravel

Laravel je open-source framework programovacího jazyku PHP. Řídí se MVC (model-view-controller) architekturou. MVC architektura dělí aplikaci na tři hlavní komponenty: modely, pohledy a kontroléry. Model reprezentuje data a logiku aplikace. Jeho úkolem je určit, jak jsou data strukturována a jaké operace je možné s nimi vykonat. Má zodpovědnost za načítání a ukládání dat z a do databáze nebo jiného datového zdroje. View (česky pohled) zodpovídá za prezentaci uživatelského rozhraní aplikace. Formátuje a zobrazuje uživateli data získané z modelu a umožňuje interakci. Kontrolér působí jako prostředník mezi modelem a pohledem. Z pohledu přijímá data, zpracovává je a podle nich upravuje model. Tohle funguje oběma směry, podle dat získaných z modelu rozhoduje, jaký pohled by měl být uživateli prezentován. Laravel poskytuje abstraktní databázovou vrstvu zvanou Eloquent ORM (Object-Relational Mapping). Tato vrstva zprostředkovává jednoduchou a efektivní komunikaci s databázemi s využitím čitelné a lehce srozumitelné syntaxe. Laravel také poskytuje zabudovaný systém ověřování, který usnadňuje implementaci uživatelské registrace, přihlášení a resetování hesla. Přestože je Laravel back-end framework, umožňuje

vývojáři vytvořit celou aplikaci včetně front-endu. Existují dvě řešení, jak vyřešit vývoj front-endu s použitím tohoto frameworku. Jedním z nich je využití funkcí Laravelu jako pohledů, šablonového enginu a frameworku Laravel Livewire. Druhým způsobem je využití JavaScript frameworků VueJS nebo ReactJS. Spojení Laravelu s těmito frameworky je komplikované a přináší spoustu problémů, naštěstí existuje nástroj Inertia, který funguje jako most mezi back-endem a VueJS nebo ReactJS frameworky. Inertia funguje s jakýmkoliv back-end frameworkem ale je vyladěn především pro Laravel. Laravel. Obsahuje vlastní CLI s názvem Artisan.

6.2 Django

Django je open-source framework programovacího jazyku Python určený pro vývoj webových aplikací. Používá MVC architekturu a je vhodný pro práci na malé, velké ale i podnikové úrovní. Obsahuje ORM, který dovoluje vývojáři interagovat s databázovými systémy jako MySQL, PostgreSQL SQLite a Oracle pomocí Python kódu. Django má zabudovaný šablonový engine, který odděluje prezentační logiku od byznysu logiky aplikace. Šablony jsou psané v Django šablonovým jazyku, který přináší značky, a filtry pro dynamické generování HTML. Za použití vbudovaného API pro zpracování formulářů, umožňuje definovat formuláře použitím tříd Pythonu a jejich následné vykreslení do šablon. Tohle usnadňuje proces manipulace s HTML formuláři a jejich ověření. Užitečnou vlastností Django frameworku je automatická generace administračního rozhraní na základě modelů dané aplikace. Toto rozhraní dovoluje ověřeným uživatelům spravovat a manipulovat s daty v databázi bez potřeby psát vlastní pohledy či šablony. Kromě již zmíněného API pro zpracování formulářů, Django také obsahuje ochranu proti běžným útokům na web jako XSS nebo SQL injekci.

6.3 Flask

Flask je dalším populárním open-source frameworkem programovacího jazyku Python určený pro vývoj webových aplikací. Na rozdíl od Djanga, Flask nepotřebuje pro spuštění žádné konkrétní nástroje nebo knihovnu. Neobsahuje žádné komponenty navíc jako abstraktní databázovou vrstvu nebo ověření formuláře. Všechny tyto věci je možné přidat pomocí rozšíření třetí strany. Flask je navržen, aby bylo snadné a rychlé s ním začít. Přesto je možné rozšiřovat a vytvářet komplexní aplikace. *„Flask vyniká mezi ostatními frameworky, protože nechá své vývojáře chopit se řízení a mít kompletní kreativní kontrolu*

nad jejich aplikacemi.“ „Možná už jste slyšeli frázi „boj s frameworkem“. Toto se děje s většinou frameworků, když se rozhodnete vyřešit problém pomocí řešení, které není oficiální. Flask takový není. Máte rádi relační databáze? Super. Flask je všechny podporuje. Možná preferujete NoSQL databáze. Žádný problém. Flask s nimi pracuje také. Chcete použít váš vlastní domácí databázový Engine? Nepotřebujete databázi? Pořád v pohodě. S Flaskem si můžete zvolit komponenty své aplikace nebo dokonce napsat své vlastní, pokud je to to, co chcete. Žádné otázky!„ [40](Vlastní překlad). Přichází se šablonovým enginem JinJa2. Pomocí tohoto enginu může vývojář vytvářet HTML šablony se zástupci pro dynamický obsah. Tyto šablony jsou poté využívané k vytváření HTML odpovědí, které se odesílají nazpět klientovi. Flask je postaven na Werkzeug. Jedná se o komplexní knihovnu nástrojů WSGI (Web Server Gateway Interface) česky rozhraní brány webového serveru a poskytuje vývojáři dobrý základ pro zpracování úkolů spojenými s HTTP.

6.4 Ruby on Rails

Ruby on Rails často také označován pouze jako Rails je open-source framework programovacího jazyku Ruby, založený na MVC architektuře. „*Rails je full-stack framework. Obsahuje všechny nástroje potřebné k vytváření úžasných webových aplikací na přední i zadní straně.“*[43] (Vlastní překlad). K těmto nástrojům patří Active Record vzor, poskytující ORM vrstvu. Scaffolding, pro automatickou generaci sady souborů MVC na základě databázového schématu. Testovací frameworky jako RSpec nebo MiniTest, podporující psaní automatických testů pro zajištění stability aplikace. Mít vše k dispozici může být skvělé při vývoji velkých projektů, ale u malých projektů může být většina těchto nástrojů úplně zbytečná.

6.5 Spring

Spring je open-source událostmi řízený framework programovacího jazyku Java. Jádrem Spring frameworku jsou IoC (Inversion of Control) a DI (Dependency Injection). Společně spravují tvorbu a konfiguraci objektů a jejich závislosti. Objekty nejsou zodpovědné za tvorbu svých závislostí, místo toho jsou do nich vloženy při běhu kódu. Spring je postaven na AOP (Aspect-Oriented Programming) česky aspektově orientované programování programovacím paradigmatu, které umožňuje modularizaci průřezových problémů jako je protokolování, zabezpečení a správa transakcí. Díky podpoře AOP, Spring umožňuje vývojáři oddělit tyto problémy od byznysu logiky aplikace. Spring nabízí podporu různých

technologií pro přístup k datům včetně ORM frameworků jako Hibernate a JPA (Java Persistence API) a databází NoSQL. Poskytuje konzistentní a zjednodušený přístup k práci s databázemi, čímž omezuje množství šablonovitého kódu. Podobně jako Rails, Spring nabízí možnost integrace testovacích frameworků jako jsou JUnit a Mockito. Skvělou vlastností je možnost bezproblémové integrace s ostatními frameworky a technologiemi jako Java EE, Struts nebo Apache Tiles. Spring se dělí na tři moduly Spring Framework, Spring Boot a Spring Cloud a dovoluje vývojáři si zvolit, který z nich chce využít. Spring Framework je základ celého frameworku a obsahuje funkce a nástroje využívané všemi projekty co nesou jméno Spring. Spring Boots přináší princip konvence nad konfigurací. To znamená, že sebou nese základní nastavení na základě konvencí, aby usnadnil nebo odstranil potřebu konfigurace. Spring Cloud je set nástrojů a knihoven pro vývoj aplikací využívajících cloud computingu.

7 Pokročilá témata webových technologií

7.1 AJAX

AJAX (Asynchronous JavaScript and XML) česky Asynchronní JavaScript a XML. „*Přestože X ve zkratce AJAX znamená XML, JSON je preferovaný, protože je menší a je psán v JavaScriptu. Obojí JSON a XML jsou používány pro zabalení informací v AJAX modelu*“.[47] (Vlastní překlad). Asynchronní znamená, že webová aplikace vyměňuje data se serverem na pozadí aplikace bez potřeby obnovit celou stránku. JavaScript odešle asynchronní žádost pomocí objektu XMLHttpRequest na server. Server žádost zpracuje a odešle odpověď klientovi. JavaScript zpracuje odpověď a dynamicky aktualizuje stránku bez jejího obnovení. AJAX není knihovna jazyk ani framework, jedná se o techniku pro tvorbu webu. Pomocí této techniky je možné používat technologie jako například ty zmíněné v této práci společně.

7.2 jQuery

jQuery je rychlá, malá a funkčně bohatá knihovna jazyka JavaScript. Díky snadno použitelnému rozhraní API, které funguje v mnoha prohlížečích, usnadňuje například procházení a manipulaci s dokumenty HTML, zpracování událostí, animace a Ajax. Díky kombinaci všestrannosti a rozšiřitelnosti změnila knihovna jQuery způsob, jakým miliony lidí píší JavaScript.[48](Vlastní překlad)

7.3 JSON

JSON (JavaScript Object Notation) česky Zápis objektů v jazyce JavaScript. JSON je formát pro reprezentaci dat, je nenáročný a poskytuje jednoduché čtení a zapisování dat. Což je jedním z hlavních důvodů proč pro většinu vývojářů nahradil formát XML. Dalším důvodem je velikost souboru, JSON soubory jsou menší v porovnání s ostatními formáty. Mezi podporované typy dat patří: String, Čísla, Boolean, Pole, Objekty a Null. Syntaxe se skládá z klíčových hodnot uzavřených do složených závorek a oddělených čárkami. JSON formát je navržen, aby byl jednoduše čitelný a zapisovatelný člověkem. Je snadné jej integrovat s většinou programovacích jazyků.

7.4 CMS

CMS (Content Management Systém) česky systém pro správu obsahu. CMS vzniknul, aby web mohl spravovat kdokoliv s nižší úrovní znalostí a technologie se více rozšířila. Jedná se o software, který umožňuje vytvořit a spravovat obsah ve spolupráci s dalšími uživateli. CMS platformy mají vlastní uživatelsky přístupné rozhraní, dovolující jednotlivcům a organizacím jednoduše vytvořit a aktualizovat webové stránky, blogy, online obchody a jiné digitální platformy. K tomuto účelu lze využít nástroje pro tvorbu a úpravu obsahu jako jsou textové a obrázkové editory, nahrávače videí a formátování. Populárními CMS platformami jsou WordPress, Joomla, Drupal a Magento. Technologie, které se používají v kombinaci s CMS, jsou PHP, Python, Ruby, MySQL, PostgreSQL, SQLite, HTML/CSS, JavasScript a Bootstrap. Knihovny jako Laravel, Django, Ruby on Rails se také často používají v kombinaci s CMS.

7.5 API

API (Application Programming Interface) česky Rozhraní pro programování aplikací. Je to sada pravidel a protokolů, které dovolují různým aplikacím mezi sebou komunikovat. Definují, jak by spolu měly různé softwarové komponenty interagovat a určuje metody, datové formáty a protokoly použité pro komunikaci. API hrají důležitou roli ve vývoji moderních webových aplikacím, jelikož umožňují vývojářům vytvářet aplikace, které využívají funkce jiných aplikací, bez potřeby hlubších znalostí o jejich fungování.

7.6 Ukázka využití AJAX techniky

7.6.1 HTML

V následující ukázce bude předvedeno, jak lze využít AJAX techniky v kombinaci s HTML, CSS, JavaScript a PHP jazyky pro tvorbu komentářové sekce na webové stránce.

```
401     <!-- Komentáře -->
402     <h1>Komentářová sekce</h1>
403     <div class="wrapper">
404       <form id="form">
405         <div class="inputBox">
406           <label for="name"> <strong> Jméno: </strong></label>
407           <br>
408           <input type="text" id="name" placeholder="Zadej své jméno" required>
409         </div>
410         <div class="inputBox">
411           <label for="msg"> <Strong> Komentář: </Strong> </label>
412           <br>
413           <textarea id="msg" placeholder="Napiš komentář" required></textarea>
414         </div>
415         <button id="btn">Odeslat</button>
416       </form>
417       <hr>
418       <div class="content" id="content">
419
420       </div>
421     </div>
422   </div>
```

Obrázek 14: HTML – komentáře (David Grydil)

Pomocí HTML je vytvořena struktura komentářové sekce. Celá sekce je obalena třídou *„wrapper“*. Uvnitř třídy je element *<form>*, který je využit ke sběru vstupů uživatele. Tyto vstupy jsou vytvořeny pomocí značek *<input>* pro jméno autora komentáře a *<textarea>* pro samotný komentář. Atribut *<required>* dělá vyplnění vstupu pro komentář povinné. Kód poté pokračuje vytvořením tlačítka, na které bude později navázána funkce pro odeslání komentáře a končí třídou *„content“*, která bude použita pro zobrazení komentářů.

7.6.2 JavaScript, AJAX, jQuery

Druhá část kódu využívá jQuery knihovny k provádění asynchronních HTPP požadavků (AJAX) a zpracování odesílání formulářů.

```
<!-- jQuery knihovna -->
<script src="https://code.jquery.com/jquery-3.7.0.min.js"></script>
<script>
  $(document).ready(function () {
    function loadData() {
      $.ajax({
        url: 'select-data.php',
        type: 'POST',
        success: function (data) {
          $("#content").html(data);
        }
      });
    }
    loadData();

    $("#btn").on("click", function (e) {
      e.preventDefault();
      var name = $("#name").val();
      var msg = $("#msg").val();

      $.ajax({
        url: 'insert-data.php',
        type: 'POST',
        data: { name: name, msg: msg },
        success: function (data) {
          if (data == 1) {
            loadData();
            alert('Komentář úspěšně odeslán');
            $("#form").trigger("reset");
          } else {
            alert('Komentář nebyl odeslán');
          }
        }
      });
    });
  });
</script>
```

Obrázek 15: AJAX – komentáře (David Grydil)

Kód je obalen do funkce *$ (document).ready()*. Tím je zajištěno, že se kód uvnitř této funkce provede až po dokončení načítání DOM. Uvnitř této funkce je vnořená funkce *loadData()*. Tato funkce je zodpovědná za provedení AJAX požadavku na server pro načtení dat z koncového bodu *select-data.php*. AJAX požadavek se provádí jQuery metodou *$.ajax()*, která je pro provádění asynchronních HTTP požadavků. Požadavek je nakonfigurován s následujícími vlastnostmi:

- *url*: Určující adresu URL skriptu na straně serveru, který bude požadavek zpracovávat, v tomto případě „select-data.php“.
- *type*: Určující HTTP metody, která bude použita pro požadavek, v tomto případě *POST*.
- *success*: Určující funkci zpětného volání, která se provede v případě úspěchu požadavku. Přijatá data jsou této funkci předána jako argument. V tomto případě jsou přijadá data vložena do HTML elementu s id *„content“* pomocí příkazů *$("#content").html(data)*.

Po definování funkce „loadData()“ se tato funkce ihned zavolá pomocí funkce „loadData()“. Tím je zajištěno, že se data přivedou a načtou při načtení stránky. Dále je zde obsluha události, která je vázaná na událost kliknutí na tlačítko s id „btn“. Obsluha události je definována pomocí *$("#btn").on("click", function (e) { ... })*, kde funkce přijímá jako

argument objekt události *e*. Uvnitř funkce obsluhy události je příkaz *e.preventDefault(),* který zabraňuje výchozímu chování při odeslání formuláře. Zabraňuje opětovnému načtení stránky, které by nastalo při zpracování odeslání formuláře prostřednictvím techniky AJAX. Kromě tohoto příkazu je zde také funkce *$("[id]").val()*, která získává hodnoty ze dvou vstupních polí s id *name* a *msg*. Tyto hodnoty se ukládají do proměnných „*name*“ a *msg*.

Poté se provede další AJAX požadavek, tentokrát na koncový bod *insert-data.php*

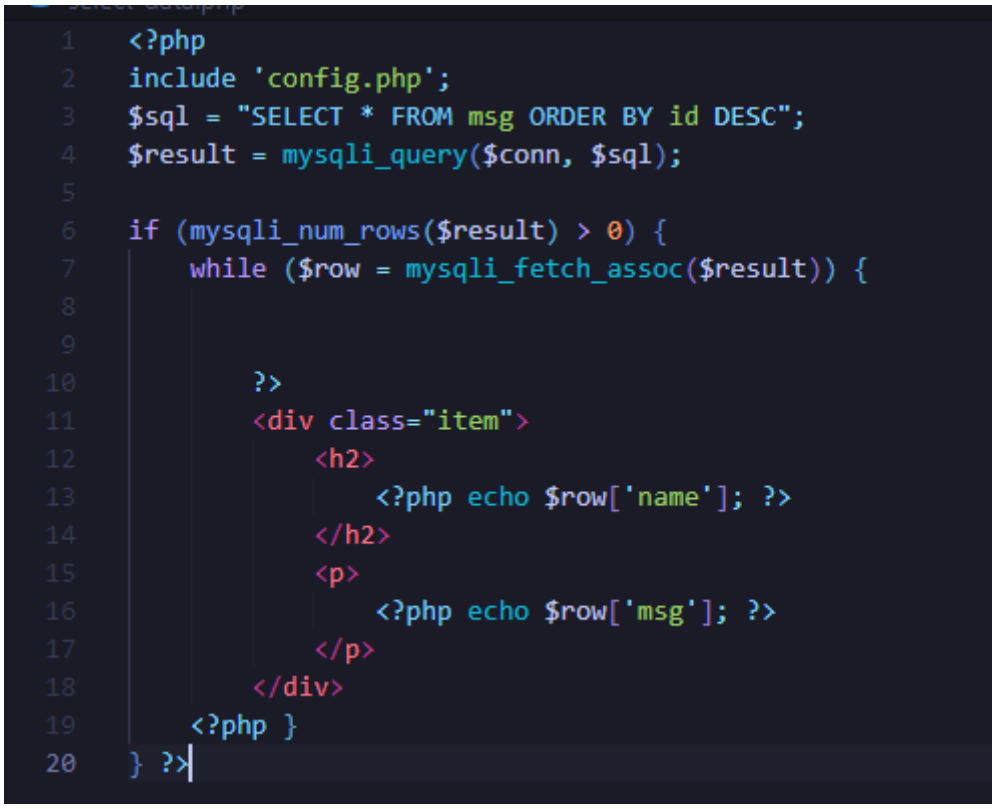
Tento požadavek je nakonfigurován s vlastnostmi:

- *url*: Určující adresu URL skriptu na straně serveru, který bude požadavek zpracovávat, v tomto případě *insert-data.php*.
- *type*: Určující HTTP metody, která bude použita pro požadavek, v tomto případě *POST*.
- *data*: Odesílající hodnoty *name* a *msg* na server.

Funkce zpětného volání *success* je definována pro zpracování odpovědi ze serveru. Pokud se přijatá data rovnají 1 zavolá se funkce *loadData()* pro obnovení dat, zobrazí se upozornění a formulář se resetuje pomocí funkce *("#form").trigger("reset")*. V opačném případě, tedy kdy se data nerovnají 1, se zobrazí chybové upozornění.

7.6.3 Ukázka PHP

Byly zmíněny koncové body *select-data.php* a *insert-data.php*, jedná se o soubory, ve kterých je vepsán PHP kód pro přivedení dat z databáze umístěn v souboru *select-data.php* a kód pro vložení dat do databáze v souboru *insert-data.php*. Oba soubory se připojí k databázi, pomocí propojení se souborem *config.php*.



```php
<?php
include 'config.php';
$sql = "SELECT * FROM msg ORDER BY id DESC";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    while ($row = mysqli_fetch_assoc($result)) {


        ?>
        <div class="item">
            <h2>
                <?php echo $row['name']; ?>
            </h2>
            <p>
                <?php echo $row['msg']; ?>
            </p>
        </div>
    <?php }
} ?>
```

Obrázek 16: PHP select-data (David Grydil)

Obrázek 16 ukazuje PHP kód zapsaný v souboru *select-data.php*. Kód začíná vložením souboru *config.php*, který obsahuje nezbytné informace pro navázání spojení s databází. SQL dotaz, který je zapsán v proměnné *.$sql* získá všechny řádky z tabulky *msg* a seřadí je sestupně podle sloupce *id*. K provedení SQL dotazu se použije funkce *mysqli_query()*, která přijímá dva argumenty: objekt připojení k databázi *$conn* a SQL dotaz *$sql*. Výsledek provedení dotazu je uložen v proměnné *$result*. Kód kontroluje, jestli je počet řádků vrácených dotazem *mysqli_num_rows($result)* větší než 0. Pokud ano, řádky jsou vráceny a kód vstoupí do smyčky *while*. Uvnitř této smyčky se použije funkce *myslqi_fetch_assoc()* k načtení každého řádku ze sady výsledků jako asociativního pole. Proměnné *$row* jsou při každé iteraci smyčky přiřazena data aktuálního řádku. Poté je PHP kód dočasně uzavřen pomocí *?>*, pro umožnění zápisu HTML značky přímo v rámci smyčky. Uvnitř smyčky je vytvořena třída *item*, která zobrazuje každý řádek dat z tabulky *msg*. K datům z aktuálního řádku se přistupuje pomocí proměnné *$row* a klíče asociativního pole odpovídajících názvům sloupců. V tomto případě se přistupuje ke sloupcům *name* a *msg*. Získaná data se pak zobrazí v rámci značky HTML pomocí příkazů PHP „echo“. V PHP kódu se pokračuje pomocí *<?php* aby se vstoupilo do kontextu PHP. Smyčka *while* tedy pokračuje, dokud nejsou zpracovány všechny řádky ze sady výsledků.

```php
<?php
include 'config.php';

$name = $_POST['name'];
$msg = $_POST['msg'];
$sql = "INSERT INTO msg (name, msg) VALUES ('$name','$msg')";
$result = mysqli_query($conn, $sql);

if ($result) {
    echo 1;
} else {
    echo 0;
}
?>
```

Obrázek 17: PHP insert-data (David Grydil)

Další obrázek, tedy Obrázek 17 ukazuje kód ze souboru *insert-data.php*. Kód opět začíná vložením souboru *config.php* aby navázal spojení s databází. Poté načte hodnoty parametrů *name* a *msg* z požadavku *POST* pomocí super globálu *$_POST*. Tyto hodnoty se přiřadí proměnným *$name* a *$msg*. SQL dotaz, který je definován jako řetězec v proměnné *$sql* vkládá pomocí příkazu *INSERT* přijaté hodnoty name a *msg* do tabulky *msg*. K provedení SQL dotazu je použita funkce *mysqli_query()*, která přijímá dva argumenty, objekt připojení k databázi *$conn* a SQL dotaz *$sql*. Výsledek provedení dotazu je uložen v proměnné *$result*. Kód poté kontroluje hodnotu proměnné *$result*, aby určil úspěšnost operace. Pokud je hodnota proměnné *$result* pravdivá, což nastane v případě, že operace s databází proběhla úspěšně, kód vypíše jako odpověď 1. Pokud je hodnota této proměnné nepravdivá neboli operace se nezdařila, kód jako odpověď vypíše 0.

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$database = "comment_system_ajax";

$conn = mysqli_connect($servername, $username, $password, $database);

if (!$conn) {
    echo "Spojení selhalo.";
}
?>
```

Obrázek 18: PHP – navázání spojení s databází (David Grydil)

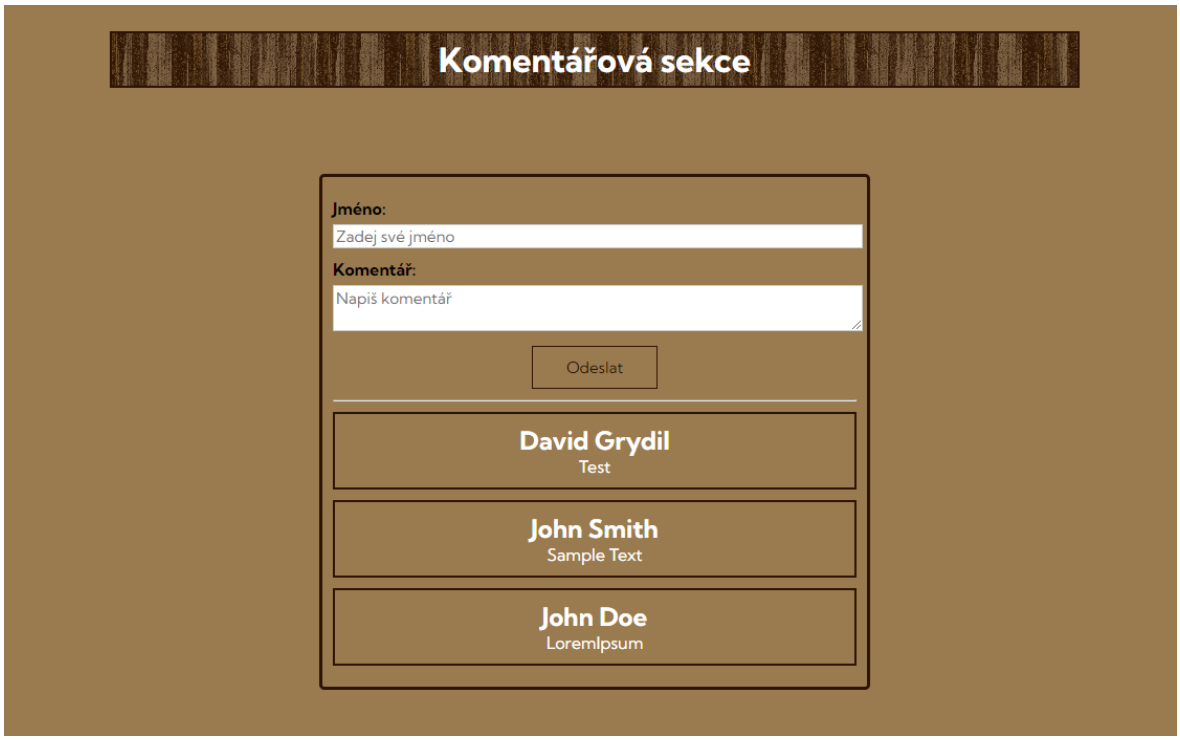
Obrázek 18 ukazuje PHP kód k navázání spojení s MySQL databázi. Kód začíná přiřazením hodnot proměnným *$servername*, *$username*, *$password* a „$database“. Tyto proměnné jsou potřebné pro prověření při připojení k databází MySQL. Jelikož databázový server běží na místním počítači, proměnná *$servername* je *localhost*, *$username* je *root*, $password neexistuje a *$database* je název databáze v tomto případě tedy *comment_system_ajax*. Kód používá funkci *mysli_connect()* k navázání spojení s databází MySQL. Jako argumenty přijímá výše zmíněné proměnné. Pokud je připojení k databázi úspěšné, funkce *mysqli_connect()* vrátí objekt připojení k databází MySQL, který je uložen v proměnné *$conn*. Tento objekt použit pro další operace s databází. Pokud se připojení nezdaří, kód vstoupí do příkazu *if* a vypíše: *Spojení selhalo.*.

7.6.4 MySQL tabulka

Struktura tabulky databáze použitá v ukázce vypadá takhle.

Obrázek 19: MySQL – tabulka (David Grydil)

7.6.5 Komentářová sekce



Obrázek 20: Komentářová sekce

Závěr

Závěrem lze říci, že moderní web je výsledkem neustálých inovací a zavádění nejmodernějších webových technologií. Prostřednictvím této práce jsme prozkoumali základní prvky a pokročilé rámce, které utvářejí tvorbu moderního webu. Studie zdůraznila význam HTML5, CSS3 a JavaScriptu při vytváření vizuálně přitažlivých a interaktivních webových stránek. Kromě toho jsme pronikli do významu front-endových frameworků jako Angular, ReactJS a VueJS, které umožňují vývoj dynamických a škálovatelných webových aplikací.

Dále byla v práci zdůrazněna úloha technologií na straně serveru, jako jsou PHP, Python, Ruby, Java, SQL a jejich frameworků Laravel, Django, Flask, Ruby on Rails, Spring a NodeJS pro efektivní zpracování dat a vytváření robustních back-endových architektur. Diskutována byla také integrace AJAX techniky a její využití při tvorbě interaktivního a dynamického webu.

Výzkum provedený v této práci poukazuje na zásadní roli moderních webových technologií při utváření digitálního prostředí. Využitím těchto technologií mohou vývojáři a designéři vytvářet webové aplikace, které splňují vyvíjející se potřeby uživatelů a poskytují poutavé a uživatelsky přívětivé zážitky. Pro udržení náskoku v neustále se měnícím digitálním světě je však nezbytné neustále se přizpůsobovat novým technologiím a trendům v oboru.

Tato práce tedy shrnula informace o jednotlivých technologiích a jejich místě jak v životě uživatele, tak v životě webového vývojáře. Přijetím těchto technologií a využitím jejich potenciálu můžeme řídit budoucnost vývoje webových aplikací a vytvářet inovativní a působivé webové zážitky, které obohatí životy uživatelů po celém světě.

Zdroje:

[1] NAIK, Umesha a D. SHIVALINGAIAH. *Comparative Study of Web 1.0, Web 2.0 and Web 3.0* [online]. In: March 2009, [cit. 2023-05-24]. Dostupné z: doi:10.13140/2.1.2287.2961

[2] MADURAI, Vivek. Web Evolution from 1.0 to 3.0. In: *Medium* [online]. 17 Feb 2018 [cit. 2023-05-24]. Dostupné z: https://medium.com/@vivekmadurai/web-evolution-from-1-0-to-3-0-e84f2c06739

[3] DESAI, Padmashri. *PROGRESSION OF WEB 3.0(SEMANTIC WEB) FROM WEB 1.0: A SURVEY* [online]. Department of Electronics and Communication Engineering, SRM University, April 2009, [cit. 2023-05-25]. Dostupné z: https://www.researchgate.net/publication/305443181

[4] palaksinghal9903. World Wide Web (WWW). In: *GeeksforGeeks* [online]. 12 Feb 2023 [cit. 2023-05-25]. Dostupné z: https://www.geeksforgeeks.org/world-wide-web-www/

[5] Web Developer. *BrainStation* [online]. 2023 [cit. 2023-05-25]. Dostupné z: https://brainstation.io/career-guides/web-developer

[6] ALMUTTAIRI, Dr. Rafah M. Web Application Development: What's the Difference Between the FrontEnd and Back-End?. In: *University of Babylon* [online]. [cit. 2023-05-25]. Dostupné z: https://www.uobabylon.edu.iq/eprints/publication_4_27425_1402.pdf

[7] Web Development. In: *GeeksforGeeks* [online]. Feb 2023 [cit. 2023-05-25]. Dostupné z: https://www.geeksforgeeks.org/web-development/

[8] GOURLEY, David, Brian TOTTY, Marjorie SAYER, Anshu AGGARWAL a Sailu REDDY. *HTTP: The Definitive Guide*. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, 2002. ISBN 9781565925090.

[9] OLUWATOSIN, Haroon Shakirat. Client-Server Model. *Journal of Computer Engineering* [online]. 2014, 57-71 [cit. 2023-05-25]. ISSN 2278-0661. Dostupné z: doi:10.9790/0661-16195771

[10] ALI, Sawsan, Rana ALAULDEEN, Ali RUAA a Ruaa Ali KHAMEES. What is Client-Server System: Architecture, Issues and Challenge of Client-Server System

(Review). *Recent Trends in Cloud Computing and Web Engineering* [online]. February 2020, (2) [cit. 2023-05-25]. Dostupné z: doi:10.5281/zenodo.3673071

[11] FATIMA, Nida. A Quick Overview of Different Types of Databases. In: *Astera* [online]. June 11th, 2019 [cit. 2023-05-25]. Dostupné z: https://www.astera.com/type/blog/a-quick-overview-of-different-types-of-databases/

[12] MDN contributors. HTML: HyperText Markup Language. In: *MDN Web Docs* [online]. [cit. 2023-05-26]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML

[13] HTML. In: *GeeksforGeeks* [online]. 2023 [cit. 2023-05-26]. Dostupné z: https://www.geeksforgeeks.org/html/

[14] LAURENČÍK, Marek a Michal BUREŠ. *Tvorba www stránek v HTML a CSS*. Dotisk 2022. Praha: Grada Publishing, 2019. ISBN 978-80-271-2241-7.

[15] CSS Tutorial: CSS Introduction. In: *GeeksforGeeks* [online]. b.r. [cit. 2023-05-26]. Dostupné z: https://www.w3schools.com/css/css_intro.asp

[16] , MDN Contributors. *CSS: Cascading Style Sheets* [online]. In: . Apr 16, 2023 [cit. 2023-05-26]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS

[17] DEED, Ian. Pros and Cons of JavaScript Development. In: *Pangea* [online]. 25.01.2023 [cit. 2023-05-26]. Dostupné z: https://www.pangea.ai/dev-javascript-resources/best-practices/

[18] , MDN contributors. JavaScript. In: *MDN Web Docs* [online]. [cit. 2023-05-26]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[19] ROBIE, Jonathan, ed. What is the Document Object Model?. In: *W3C (World Wide Web Consortium)* [online]. [cit. 2023-05-26]. Dostupné z: https://www.w3.org/TR/WD-DOM/introduction.html

[20] JavaScript HTML DOM. In: *W3Schools* [online]. [cit. 2023-05-26]. Dostupné z: https://www.w3schools.com/js/js_htmldom.asp

[21] *Angular* [online]. [cit. 2023-05-26]. Dostupné z: https://angular.io

[22] *React: The library for web and native user interfaces* [online]. ©2023 [cit. 2023-05-26]. Dostupné z: https://react.dev

[23] *React: A JavaScript library for building user interfaces* [online]. Meta Platforms, ©2023 [cit. 2023-05-26]. Dostupné z: https://legacy.reactjs.org

[24] *Vue.js: The Progressive JavaScript Framework* [online]. [cit. 2023-05-26]. Dostupné z: https://vuejs.org

[25] *PHP: Hypertext Preprocessor* [online]. [cit. 2023-05-26]. Dostupné z: https://www.php.net

[26] *Python 3.11.3 documentation* [online]. [cit. 2023-05-26]. Dostupné z: https://docs.python.org/3/

[27] *Ruby: about Ruby* [online]. [cit. 2023-05-26]. Dostupné z: https://www.ruby-lang.org/en/about/

[28] *Ruby: documentation* [online]. [cit. 2023-05-26]. Dostupné z: https://www.ruby-lang.org/en/documentation/

[29] ILYUKHA, Vitaliy. Ruby vs Python: Difference Between the Programming Languages. In: *Jelvix* [online]. ©2023 [cit. 2023-05-26]. Dostupné z: https://jelvix.com/blog/red-or-blue-pill-neo-ruby-vs-python-which-will-you-choose-for-your-backend

[30] *Java: The Destination for Java Developers* [online]. © 2023 [cit. 2023-05-26]. Dostupné z: https://dev.java

[31] Java Tutorial. In: *GeeksforGeeks* [online]. 16 Apr, 2023 [cit. 2023-05-27]. Dostupné z: https://www.geeksforgeeks.org/java/?ref=shm

[32] CHAMBERLIN, Donald D. Early History of SQL. *IEEE Annals of the History of Computing* [online]. 2012, 21 November 2012, **34**(4), 78-82 [cit. 2023-05-27]. ISSN 1934-1547. Dostupné z: doi:10.1109/MAHC.2012.61

[33] *What Is SQL (Structured Query Language)?* [online]. Amazon Web Services, [cit. 2023-05-27]. Dostupné z: https://aws.amazon.com/what-is/sql/

[34] *Node.js: About Node.js* [online]. [cit. 2023-05-27]. Dostupné z: https://nodejs.org/en/about

[35] ILYUKHA, Vitaliy. *Ruby on Rails vs. Node.js: Which One Is the Best Solution?* [online]. In: . [cit. 2023-05-27]. Dostupné z: https://jelvix.com/blog/ruby-on-rails-vs-node-js

[36] *Laravel: The PHP Framework for Web Artisans* [online]. [cit. 2023-05-27]. Dostupné z: https://laravel.com

[37] *Inertia: THE MODERN MONOLITH* [online]. [cit. 2023-05-27]. Dostupné z: https://inertiajs.com

[38] *Django: The web framework for perfectionists with deadlines* [online]. [cit. 2023-05-27]. Dostupné z: https://www.djangoproject.com/start/overview/

[39] GUPTA, Aryan. All You Need To Know About Django Framework. In: *Simplilearn* [online]. Feb 23, 2023 [cit. 2023-05-27]. Dostupné z: https://www.simplilearn.com/tutorials/django-tutorial/what-is-django-python

[40] GRINBERG, Miguel. *Flask Web Development: developing web application with python*. Sebastopol: O'Reilly Media, 2014. ISBN 978-1-449-37262-0.

[41] *Full Stack Python: flask* [online]. [cit. 2023-05-28]. Dostupné z: https://www.fullstackpython.com/flask.html

[42] *Full Stack Python: jinja2* [online]. [cit. 2023-05-28]. Dostupné z: https://www.fullstackpython.com/flask.html

[43] *Ruby on Rails* [online]. [cit. 2023-05-28]. Dostupné z: https://rubyonrails.org

[44] *Rails Guides* [online]. [cit. 2023-05-28]. Dostupné z: https://guides.rubyonrails.org

[45] *Spring: why spring?* [online]. [cit. 2023-05-28]. Dostupné z: https://spring.io/why-spring

[46] *Spring: spring framework* [online]. [cit. 2023-05-28]. Dostupné z: https://docs.spring.io/spring-framework/reference/overview.html

[47] *MDN web docs: ajax* [online]. [cit. 2023-05-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX

[48] *JQuery: What is jQuery?* [online]. [cit. 2023-05-28]. Dostupné z: https://jquery.com

[49] *JSON: IntroducingJSON* [online]. [cit. 2023-05-28]. Dostupné z: https://www.json.org/json-en.html

[50] *MDN web docs: CMS* [online]. [cit. 2023-05-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/CMS

[51] *Amazon Web Services: What Is An API (Application Programming Interface)?* [online]. [cit. 2023-05-28]. Dostupné z: https://aws.amazon.com/what-is/api/

[52] W3C TECHNICAL ARCHITECTURE GROUP (TAG). *World Wide Web Consortium* [online]. [cit. 2023-06-02]. Dostupné z: https://www.w3.org/2001/tag/doc/web-https

[53] MICHÁLEK, Martin. *CSS: moderní layout*. [Praha]: Martin Michálek – Vzhůru dolů, [2022]. ISBN 978-80-88253-07-5.

[54] WELLING, Luke a Laura THOMSON. *Mistrovství PHP a MySQL*. Brno: Computer Press, 2017. ISBN 978-80-251-4892-1.

Zdroje použité v ukázkách

[55] DHAKAR, Lokesh. *Lokesh Dhakar: Projects - LIGHTBOX* [online]. [cit. 2023-06-08]. Dostupné z: https://lokeshdhakar.com/projects/lightbox2/

[56] *JQuery: releases* [online]. In: . [cit. 2023-06-08]. Dostupné z: https://releases.jquery.com

[57] USOLTSEV, Alexey. *TRANSPARENT TEXTURES: Wood Pattern* [online]. In: . [cit. 2023-06-08]. Dostupné z: https://www.transparenttextures.com/wood-pattern.html

[58] SHARMA, Saurabh. *Google Fonts: Kumbh Sans* [online]. In: . [cit. 2023-06-08]. Dostupné z: https://fonts.google.com/specimen/Kumbh+Sans?query=kumbh

[59] *Mikroregion Předina* [online]. © 2023 [cit. 2023-06-10]. Dostupné z: https://www.predina.cz

Anotace

Jméno a příjmení:	David Grydil
Katedra nebo ústav:	Katedra technické a informační výchovy
Vedoucí práce:	prof. PhDr. Milan Klement Ph.D.
Rok obhajoby:	2023

Název práce:	Využití webových technologií pro tvorbu moderního webu
Název práce v angličtině:	Using web technologies for modern web development
Anotace práce:	Cílem bakalářské práce je popis vybraných moderních webových technologií a jejich použití při vývoji webu. Věnuje se oběma základním konceptům vývoje moderního webu, tedy front end, back end, jejich jazykům a frameworkům. Práce obsahuje ukázku využití některých technologií a technik popsaných v teoretické části.
Klíčová slova:	WWW, HTTP, Front-end, Back-end, Fullstack, HTML, CSS, JavaScript, PHP, Python, Ruby, Java, SQL, AJAX, CMS, API
Anotace v angličtině:	The aim of the bachelor thesis is to describe selected modern web technologies and their use in web development. It focuses on the two basic concepts of modern web development, i.e., front end, back end, their languages, and frameworks. The thesis includes a demonstration of the use of some of the technologies and techniques described in the theoretical part.
Klíčová slova v angličtině:	WWW, HTTP, Front-end, Back-end, Fullstack, HTML, CSS, JavaScript, PHP, Python, Ruby, Java, SQL, AJAX, CMS, API
Abstrakt:	Rychlý vývoj webových technologií zásadně změnil digitální prostředí a umožnil vznik moderního webu. Tato práce se zabývá vybranými moderními webovými technologiemi a jejich vlivem na konstrukci a funkčnost webových aplikací. Popisem základních prvků webu, jako jsou HTML5, CSS3 a JavaScript, objasňujeme jejich roli při utváření vizuálně přitažlivých a interaktivních webových stránek. Kromě toho zkoumáme pokročilé front-endové frameworky a knihovny, včetně Angular, ReactJS a VueJS, které vývojářům umožňují vytvářet sofistikované a škálovatelné webové aplikace. Dále se zabýváme technologiemi na straně serveru, jako jsou PHP, Python, Ruby, Java, SQL a frameworky jako Laravel, Django, Flask, Ruby on Rails, Spring a NodeJS. Zkoumáme jejich místo a použití pro dosažení efektivního zpracování dat a robustní architektury back-endu. Dále se zabýváme integrací AJAX techniky nebo API, které usnadňuje bezproblémovou výměnu dat mezi aplikacemi. Díky prozkoumání těchto moderních webových technologií tato práce

	vybavuje čtenáře základním porozuměním o webových aplikacích, přínosům a výzvám, které jednotlivé technologie přináší a umožňuje tak porozumět co obnáší vývoje webové stránky, webu či aplikace.
Rozsah práce:	55
Jazyk práce:	Český Jazyk