VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná informatika

OPTIMALIZAČNÍ TECHNIKY A JEJICH DOPAD NA VÝKON WEBOVÝCH APLIKACÍ

Bakalářská práce

Autor práce: Tomáš Štveráček

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

Jihlava 2025

Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce:	**Tomáš Štveráček**
Studijní program:	Aplikovaná informatika
Obor:	Aplikovaná informatika
Garant studijního programu:	Ing. Lenka Kuklišová Pavelková, Ph.D.
Název práce:	**Optimalizační techniky a jejich dopad na výkon webových aplikací**
Vedoucí práce:	PaedDr. František Smrčka, Ph.D.
Cíl práce:	Cílem bakalářské práce je vytvořit webovou aplikaci, která poslouží jako platforma pro implementaci a testování různých optimalizačních technik. Práce se zaměří na důležité aspekty optimalizace výkonu webových aplikací, prozkoumá moderní přístupy a zanalyzuje jejich vliv na výkon. Významnou částí práce bude měření metrik prostřednictvím Core Web Vitals. Dále práce přinese porovnání a analýzu vybraných nástrojů pro měření výkonu, jako jsou PageSpeed Insights, GTmetrix, WebPageTest, Pingdom Tools nebo Chrome DevTools. Zahrnuto bude rovněž uživatelské testování, přičemž bude zkoumán dopad optimalizace jak na desktopová, tak na mobilní zařízení. V závěrečné části budou uvedena doporučení pro vývojáře a návrhy pro budoucí výzkum v této oblasti.

Abstrakt

Bakalářská práce se zaměřuje na optimalizaci výkonu webových aplikací s důrazem na metriky Core Web Vitals – Largest Contentful Paint, Cumulative Layout Shift a Interaction to Next Paint. Práce představuje optimalizační techniky zahrnující asynchronní zpracování JavaScriptu, konverzi obrázků do moderních formátů, definování rozměrů prvků pro prevenci posunů či využití cachování při práci s veřejnými API. Praktická část zahrnuje testovací aplikaci, která umožňuje měřit vliv optimalizací v reálných podmínkách přepínáním mezi optimalizovanou a neoptimalizovanou verzí. K měření jsou využívány nástroje jako PageSpeed Insights, GTmetrix a WebPageTest. Výsledky testování na různých zařízeních a rychlostech připojení ukazují výrazné zlepšení metrik po aplikaci optimalizací. Práce přináší poznatky o významu jednotlivých technik a jejich dopadu na uživatelskou zkušenost v různých podmínkách.

Klíčová slova

Core Web Vitals; webová optimalizace; React; nástroje měření výkonu; testování výkonu; uživatelská zkušenost

Abstract

The bachelor thesis focuses on optimizing the performance of web applications with emphasis on the Core Web Vitals metrics – Largest Contentful Paint, Cumulative Layout Shift and Interaction to Next Paint. The thesis presents optimization techniques including asynchronous JavaScript processing, converting images to modern formats, defining element sizes to prevent shifts, and using caching when working with public APIs. The practical part includes a test application that allows to measure the impact of optimizations in real conditions by switching between optimized and non-optimized versions. Tools such as PageSpeed Insights, GTmetrix and WebPageTest are used for measurement. Test results on different devices and connection speeds show a significant improvement in metrics after applying optimizations. The paper provides insights into the importance of each technique and its impact on user experience in different conditions.

Keywords

Core Web Vitals; web optimization; React; performance measurement tools; performance testing; user experience

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

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

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

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

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

V Jihlavě dne 1. dubna 2025

…….

Podpis studenta

Poděkování

Děkuji vedoucímu práce PaedDr. Františku Smrčkovi, Ph.D. za odborné vedení a cenné připomínky. Velký dík patří mé rodině za podporu během celého studia a partnerce za trpělivost a pochopení. Nemohu opomenout ani kvalitní kávu, jež byla věrným společníkem při dlouhých hodinách programování a psaní.

Obsah

Seznam obrázků ... **8**
Seznam tabulek ... **9**
Seznam grafů ... **10**
Seznam zkratek ... **11**
Úvod ... **12**
1 Měření výkonu webových aplikací ... **13**
1.1 Syntetické měření ... 13
1.2 Měření reálných uživatelů (RUM) ... 13
2 Přehled metrik Core Web Vitals ... **14**
2.1 Largest Contentful Paint ... 14
2.2 Cumulative Layout Shift ... 16
2.3 Interaction to Next Paint ... 17
3 Doplňkové metriky a ukazatele výkonu ... **19**
3.1 Time to First Byte ... 19
3.2 DOM Content Loaded ... 19
3.3 First Contentful Paint ... 19
3.4 Total Blocking Time ... 20
3.5 Speed Index ... 20
3.6 Lighthouse Performance Score ... 21
4 Nástroje pro měření výkonu ... **22**
4.1 PageSpeed Insights ... 22
4.2 Google Lighthouse ... 23
4.3 GTmetrix ... 24
4.4 WebPageTest ... 25
4.5 Pingdom Tools ... 27
4.6 Ostatní nástroje ... 28
5 Optimalizační techniky ... **30**
5.1 Minifikace kódu ... 30
5.2 Lazy loading ... 31
5.3 Komprese obrázků ... 31
5.4 Optimalizace API volání ... 32
5.5 Využití CDN ... 32
6 Použité technologie ... **34**
6.1 HTML ... 34
6.2 SCSS ... 34
6.3 React ... 34
6.4 Netlify ... 34
7 Návrh a implementace aplikace ... **35**
7.1 Architektura aplikace ... 35
7.2 Implementace projektu ... 37
7.3 Implementace PSI API ... 42
8 Analýza optimalizačních technik ... **45**

8.1 Metodologie testování 45
8.2 Výsledky testování optimalizací 45
8.3 Závěry z testování optimalizačních technik 51
9 Uživatelské testování různých podmínek 52
9.1 Metodika testování 52
9.2 Srovnání výkonu na různých reálných zařízeních 52
9.3 Závěry z uživatelského testování 55
Závěr 57
Seznam použité literatury 58
Přílohy 62

Seznam obrázků

Obr. 1: Výsledky měření v nástroji PageSpeed Insights 23
Obr. 2: Výsledky měření z nástroje Lighthouse 24
Obr. 3: Výsledky měření v nástroji GTmetrix 25
Obr. 4: Výsledky měření v nástroji WebPageTest 26
Obr. 5: Výsledky měření v nástroji Pingdom Tools 28
Obr. 6: Část úvodní stránky aplikace 35
Obr. 7: Implementace přepínání mezi verzemi stránky 38
Obr. 8: Monitoring Core Web Vitals metrik 39
Obr. 9: Implementace obrázků s nastavením parametrů 40
Obr. 10: Definice komponenty LayYoutubeEmbeded.jsx 40
Obr. 11: Použití komponenty LazyYoutubeEmbed v hlavní testovací stránce 41
Obr. 12: Optimalizace interaktivity asynchronním zpracováním 41
Obr. 13: Funkce pro volání PageSpeed Insights API 42
Obr. 14: Funkce pro spuštění testu 43
Obr. 15: Tlačítko a zobrazení výsledků testu 44
Obr. 16: Vliv rychlosti připojení na metriku LCP 51
Obr. 17: Vliv hardwaru na metriky LCP a CLS 56

Seznam tabulek

Tab. 1: Váhy a ideální hodnoty metrik tvořících Lighthouse Performance Score 21
Tab. 2: INP testy – Fibonacciho posloupnost a odeslání formuláře 47
Tab. 3: Rychlost načítání knihoven z různých zdrojů (první načtení vs. cachované) 48
Tab. 4: Výsledky testování výkonu vybraných API 49
Tab. 5: Hodnoty naměřené pomocí web-vitals knihovny 53
Tab. 6: Hodnoty naměřené pomocí Lighthouse 53
Tab. 7: Hodnoty INP na mobilních zařízeních 54
Tab. 8: Hodnoty INP na počítačích 54

Seznam grafů

Graf 1: LCP testy na desktopu v různých nástrojích 46
Graf 2: CLS testy pro různé typy obsahu na desktopu 47
Graf 3: Vliv rychlosti připojení na LCP 50

Seznam zkratek

API	Application Programming Interface
AVIF	AV1 Image File Format
CDN	Content Delivery Network
CLS	Cumulative Layout Shift
CSS	Cascading Style Sheets
DCL	DOM Content Loaded
DDoS	Distributed Denial of Service
FCP	First Contentful Paint
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
INP	Interaction to Next Paint
JPEG	Joint Photographic Experts Group
LCP	Largest Contentful Paint
LPS	Lighthouse Performance Score
RUM	Real User Monitoring
PNG	Portable Network Graphics
PSI	PageSpeed Insights
SASS	Syntactically Awesome Stylesheets
SCSS	Sassy CSS
SEO	Search Engine Optimization
SI	Speed Index
SVG	Scalable Vector Graphics
TBT	Total Blocking Time
TTFB	Time To First Byte
URL	Uniform Resource Locator
WebP	Web Picture format

Úvod

S neustálým rozvojem webových technologií a rostoucím počtem aplikací se zvyšují i očekávání uživatelů na jejich rychlost a plynulost. Každé zpoždění při načítání stránky může odradit zákazníka, což je zvlášť problematické v konkurenčním prostředí e-commerce platforem. Správná optimalizace výkonu je nezbytná pro udržení konkurenceschopnosti a zajištění příjemného uživatelského zážitku.

Moje motivace k volbě tématu vychází z faktu, že rychlost načítání hraje podstatnou roli nejen ve spokojenosti uživatelů, ale také optimalizaci pro vyhledávače (SEO). O optimalizaci webového výkonu se zajímám dlouhodobě a vím, že dnešní uživatelé jsou nároční a netrpěliví. Nemají rádi čekání, a už vůbec ne, když při prohlížení dochází k nepříjemným posunům obsahu kvůli pozdějšímu načítání jiných prvků, například reklam. Každý zná situaci, kdy klikne na tlačítko *odeslat*, pokud není okamžitě zřejmé, že akce proběhla, neví, jestli klikl správně. Z toho důvodu je optimalizace odezvy a výkonu webů tak důležitá.

Rovněž mě zaujal rostoucí význam metrik Core Web Vitals, které Google využívá při hodnocení kvality stránek. Metriky zahrnují Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS) a Interaction to Next Paint (INP).

Cílem práce je analyzovat a porovnat různé techniky optimalizace výkonu webových aplikací, jako jsou minifikace kódu, optimalizace obrázků, lazy loading, cache a další. Práce se zaměřuje na zkoumání způsobu, jak jednotlivé techniky ovlivňují výkon aplikací, a to jak na desktopových, tak i mobilních zařízeních.

Praktická část se zabývá vytvořením webové aplikace jako platformy pro testování různých optimalizačních technik. Aplikace umožní uživatelům dynamicky přepínat mezi jednotlivými metodami a sledovat jejich dopad na výkon. Výsledky testování jsou měřeny a analyzovány prostřednictvím nástrojů jako PageSpeed Insights, Lighthouse, WebPageTest, GTmetrix a knihovny web-vitals.

1 Měření výkonu webových aplikací

Porozumění výkonu webových stránek pomáhá zlepšit jejich rychlost a použitelnost. K tomu máme dva hlavní přístupy – syntetické měření a měření na reálných uživatelích, známé jako RUM (Real User Monitoring).

Syntetické měření probíhá v kontrolovaných podmínkách, díky čemuž je ideální pro sledování změn během vývoje a odhalování chyb. RUM naopak sbírá data od skutečných uživatelů, kteří web navštěvují, a ukazuje, jak funguje v reálném provozu. Syntetické měření je rychlé a snadno použitelné, zatímco RUM nabízí detailní pohled na uživatelskou zkušenost. Společné využití obou přístupů přináší nejpřesnější výsledky (MDN Web Docs, 2024).

1.1 Syntetické měření

Syntetické měření, často označované jako aktivní monitorování, využívá simulaci uživatelských interakcí k testování výkonu webových stránek v kontrolovaných podmínkách. Přístup aktivního monitorování odhaluje problémy ještě před tím, než ovlivní reálné uživatele. Umožňuje analyzovat výkon z různých geografických lokalit a zároveň porovnávat rychlost webu s konkurenčními stránkami. Neposkytuje pohled na skutečné podmínky, ale je užitečný pro testování nových funkcí a sledování změn. (Hansa, 2024).

K měření výkonu webových stránek slouží například nástroje jako Google PageSpeed Insights, PageSpeed.cz, WebPageTest, Google Lighthouse v Chrome DevTools či GTmetrix. Vzhledem k jednoduchosti a přehlednosti výsledků se pro začátek doporučuje PageSpeed Insights. Pokud skóre na desktopu i mobilu dosáhne alespoň 80 bodů, je vhodné se zaměřit na pokročilejší nástroje. Nejdůležitější je nejprve vyřešit problémy zvýrazněné červeně, protože výrazně ovlivňují výkon (Michálek, 2016).

1.2 Měření reálných uživatelů (RUM)

Měření reálných uživatelů (RUM – Real User Monitoring) představuje pasivní přístup ke sledování výkonu webových stránek. Data jsou získávána přímo od návštěvníků, což umožňuje zaznamenávat skutečné podmínky, jako výkon zařízení, rychlost připojení, přítomnost reklamních blokátorů či konkrétní uživatelské interakce. Pomocí RUM měření se laboratorní testování doplňuje o reálné scénáře, které by v kontrolovaném prostředí mohly zůstat přehlédnuty. RUM lze rovněž implementovat nasazením skriptu pro sběr metrik, například Core Web Vitals, a tím získávat data přímo v prostředí uživatelů (Hansa, 2024).

Jedním z nejznámějších nástrojů pro monitorování reálných uživatelů je Chrome UX Report (CrUX). Jde o databázi s daty o uživatelské zkušenosti z prohlížeče Google Chrome. Databáze je uložená v systému BigQuery a je volně dostupná všem uživatelům. Data zahrnují měřené hodnoty výkonu, například Core Web Vitals, a poskytují informace o tom, jak webové stránky fungují v reálném prostředí. Vývojáři mohou data z CrUX analyzovat pomocí nástrojů jako Google PageSpeed Insights nebo Google Search Console. Pro podrobnější analýzu lze využít například placený nástroj SpeedCurve. Využití nástroje ovšem vyžaduje vyšší finanční náklady. Další možností monitorování je Google Analytics, který poskytuje přehled o výkonu a chování uživatelů na webu (Michálek, 2019).

2 Přehled metrik Core Web Vitals

Metriky Core Web Vitals tvoří specifickou podmnožinu širší sady Web Vitals. Zabývat by se nimi neměli jen vývojáři, ale také marketéři, majitelé firem či vedoucí týmů. Jednoduše řečeno všichni, kteří chtějí zajistit kvalitní uživatelskou zkušenost na svých webových stránkách. Metriky byly poprvé oznámeny společností Google v roce 2020. Slouží jako nástroj pro optimalizaci výkonu stránek. Přestože se sada těchto metrik postupně vyvíjí, tři nejvýznamnější se zaměřují na hlavní prvky uživatelské interakce, a to načítání hlavního obsahu, stabilitu rozložení a rychlost reakce na uživatelské vstupy.

Konkrétně jde o metriky Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS) a Interaction to Next Paint (INP), které jsou podrobněji rozebrány v následujících podkapitolách (Walton, 2020).

Core Web Vitals poskytují objektivní měřítka výkonu, která odrážejí reálnou zkušenost uživatelů. Jsou využitelná na různých typech webů – od e-shopů, přes firemní stránky, až po zpravodajské portály, kde záleží na rychlé dostupnosti informací (Raykova, 2024).

Podle studie Google až 53 % uživatelů opustí web, pokud načtení trvá déle než 3 sekundy. (Shellhammer, Neel, 2016). Taková situace může výrazně ovlivnit konverzní poměr zejména u e-shopů, kde každý ztracený návštěvník znamená přímou finanční ztrátu. Naopak weby s výbornými výsledky v Core Web Vitals získávají lepší hodnocení ve vyhledávačích, což přináší vyšší viditelnost a větší dosah stránek (Raykova, 2024).

2.1 Largest Contentful Paint

Largest Contentful Paint (LCP) představuje dobu, za kterou se načte největší prvek viditelného obsahu na stránce. Může jít o obrázek, video, textový blok, SVG nebo jiný vizuální prvek. LCP ovlivňuje, jak rychle uživatel získá přístup k hlavnímu obsahu stránky, což má přímý dopad na jeho první dojem. Google doporučuje, aby hodnota LCP nepřekročila 2,5 sekundy. Hodnoty mezi 2,5 a 4,0 sekundami signalizují potřebu optimalizace, nad 4,0 sekundy je výkon považován za špatný (Walton, Pollard, 2024).

LCP je specificky navržena tak, aby poskytovala přesnější hodnocení načítání hlavního obsahu stránky než předchozí metriky, jako jsou First Contentful Paint (FCP), First Meaningful Paint (FMP) nebo Speed Index (SI). Ostatní přístupy bývají složité a méně intuitivní. Kvůli tomu Google společně s W3C zavedl snadněji měřitelné a přesnější alternativy. LCP je rovněž důležitá pro celkové hodnocení výkonu stránky – tvoří 25 % skóre ve výsledcích Lighthouse (Walton, Pollard, 2024).

2.1.1 Příčiny pomalého načítání LCP

Mezi hlavní důvody ovlivňující vyšší hodnotu LCP patří:

- **Velké obrázky a video soubory:** Mediální prvky, které nejsou správně optimalizované (například obrázky bez komprese nebo videa ve vysokém rozlišení), zvyšují dobu načítání. Problém se často vyskytuje u stránek využívajících velké úvodní grafiky nebo bannery.
- **Pomalá doba odezvy serveru:** Pokud server reaguje pomalu, celý proces načítání obsahu se zpozdí. Důvodem může být omezená kapacita serveru nebo vysoký počet požadavků během špičkové zátěže.
- **Blokující vykreslovací zdroje:** CSS a JavaScript, které jsou nezbytné pro správné zobrazení stránky mohou, blokovat vykreslení obsahu, dokud nejsou kompletně načteny a zpracovány.
- **Načítání na straně klienta:** Jestliže je obsah generován pomocí JavaScriptu na straně klienta (například pomocí frameworku jako React), může dojít k prodlevám, než je obsah vykreslen.
- **Pomalé načítání zdrojů:** Velké soubory, které nejsou komprimovány nebo nejsou distribuovány pomocí CDN, zvyšují dobu načítání. Efektivní CDN pomáhá pomalé načítání eliminovat, protože distribuuje obsah na servery blíže uživatelům.
- **Chybné nastavení lazy loading:** Špatně implementované líné načítání může způsobit, že obrázky tvořící LCP jsou načítány až příliš pozdě. To se děje, pokud atribut *loading="lazy"* zahrnuje i klíčové obrázky, které by měly být načteny okamžitě (Toder, 2024).

2.1.2 Optimalizační techniky pro zlepšení LCP

Hodnotu LCP lze zlepšit prostřednictvím následujících metod.

- **Optimalizace obrázků:** Moderní formáty, jako WebP nebo AVIF, nabízejí vysokou kvalitu při výrazně nižší velikosti souborů oproti tradičním formátům, jako jsou JPEG nebo PNG, a proto jsou vhodné pro zlepšení výkonu.
- **Lazy loading:** Obrázky, které nejsou součástí LCP, je vhodné načítat líně pomocí atributu *loading="lazy"*. Pro klíčové obrázky, které tvoří LCP, by se však lazy loading neměl používat, aby se předešlo jejich opožděnému načítání.
- **HTML5 tag <picture> místo CSS pozadí:** Obrázky tvořící LCP by měly být načítány přímo pomocí HTML, nikoliv jako pozadí prostřednictvím vlastnosti *background-image*.
- **Preloading důležitých zdrojů:** Klíčové zdroje, jako obrázky nebo videa, by měly být načteny prioritně pomocí atributu *<link rel="preload">*. Preloading zajistí, aby byly nejdůležitější soubory pro vykreslení obsahu k dispozici prioritně.
- **Optimalizace CSS:** Minimalizace a komprese CSS snižuje velikost souborů a urychluje jejich načítání. Kritické styly by měly být načteny okamžitě (*inline* nebo pomocí *preload*), zatímco nekritické styly lze odložit pomocí atributu media.

- **Async a defer u JavaScriptu:** Atributy umožňují odložit vykonávání JavaScriptu, čímž se zkracuje doba načítání hlavního obsahu:
    - **async:** Skript se načte a spustí ihned po načtení, což je vhodné pro nezávislé skripty.
    - **defer:** Skript se načte paralelně a spustí až po načtení celého HTML dokumentu, což je vhodné pro skripty, které ovlivňují obsah stránky.
- **Využití CDN:** Content Delivery Network (CDN) snižuje vzdálenost mezi uživatelem a serverem, což zrychluje načítání obsahu. U důležitých souborů však může být lepší je hostovat přímo na vlastním serveru, aby se minimalizovala závislost na službách třetích stran (Walton, Polland, 2024).

2.2 Cumulative Layout Shift

Cumulative Layout Shift (CLS) sleduje stabilitu rozložení prvků na stránce během jejího načítání. Metrika měří každý neočekávaný posun prvků na stránce, který může narušit uživatelskou zkušenost. Typickým příkladem je situace, kdy uživatel chce kliknout na tlačítko, ale stránka se posune kvůli načítání reklamy nebo externího obsahu. V důsledku toho může uživatel kliknout na nesprávný prvek. Podobným způsobem může dojít ke ztrátě orientace na stránce při čtení textu posunutého vlivem načítání jiného obsahu (Mihajlija, 2020).

Hodnoty CLS se pohybují od 0 do 1 (bez jednotek), přičemž nižší hodnoty značí lepší stabilitu zobrazení. Google doporučuje, aby skóre do 0,1 bylo považováno za dobré a stabilní, skóre mezi 0,1 a 0,25 vyžadovalo zlepšení a hodnoty nad 0,25 byly nevyhovující a vyžadovaly optimalizaci. Na rozdíl od jiných metrik Core Web Vitals není CLS vázáno na čas, což usnadňuje jeho vyhodnocení. Metrika tvoří 15 % celkového skóre Lighthouse Performance Score (Mihajlija, 2020).

2.2.1 Výpočet CLS

CLS udává hodnotu bez jednotek, která se vypočítává na základě vzorce:

$$CLS = impact\ fraction \times distance\ fraction$$

- **Impact fraction:** Procentuální část zobrazené oblasti, která je ovlivněna posunem.
- **Distance fraction:** Poměr vzdálenosti, o kterou se prvek posunul, vůči velikosti viditelné části stránky.

Například pokud se obrázek posune o 50 % viditelné plochy a ovlivní 14 % plochy, výsledek je CLS=0,5×0,14=0,07 (Mihajlija, 2020).

2.2.2 Příčiny vysoké hodnoty CLS

Zvýšenou hodnotu CLS nejvíce ovlivňují:

- **Obrázky a videa bez definovaných rozměrů:** Pokud chybí atributy *width* a *height*, prohlížeč nemůže rezervovat prostor před načtením obsahu.

- **Reklamy a widgety třetích stran:** Dynamické reklamy, iframy a vložené prvky (například videa z YouTube nebo mapy z Google Maps) často nemají definovaný prostor, což způsobuje posuny.
- **Webová písma:** Pomalé načítání webových písem může způsobit, že prohlížeč dočasně použije systémové písmo, to vede k posunu při pozdějšímu vykreslení správného písma.
- **Animace:** Využívání CSS vlastností jako *top* a *left* v animacích způsobuje změny rozložení (Osmani, Polland, 2024).

2.2.3 Optimalizační techniky pro snížení CLS

Ke snížení CLS mohou pomoci následující kroky:

- **Nastavení rozměrů obrázků a videí:** Definování atributů *width* a *height* nebo využití CSS vlastnost *aspect-ratio*.
- **Rezervace prostoru pro reklamy a widgety:** Použít zástupné rozměry nebo pevné kontejnery, které minimalizují posuny při načítání obsahu. Pomocí CSS můžeme použít vlastnost *aspect-ratio*, případně trik s *paddingem*, jenž spočívá v nastavení vnitřního vyplnění (*padding*) na prvek pomocí procent, která odpovídají poměru stran. Například pro poměr stran 16:9 lze použít *padding-top: 56.25%*.
- **Optimalizace načítání písem:** Načíst písma pomocí *rel="preload"* a zajistit fallback písma s podobnými vlastnostmi. Případně použít atribut *font-display: swap*.
- **Animace:** Používat vlastnosti *transform* a *opacity* místo *top*, *left* nebo *margin*.
- **Použití bfcache:** Prohlížeče mohou ukládat stránky do paměti při navigaci zpět/vpřed, což zajišťuje okamžité zobrazení bez posunů. Pro zajištění kompatibility je důležité vyhnout se blokátorům, jako jsou komplexní interakce JavaScriptu nebo nestandardní navigace (Raykova, 2024).

2.3 Interaction to Next Paint

Interaction to Next Paint (INP) měří rychlost, s jakou webová stránka reaguje na uživatelskou akci. Například se jedná o akce typu jako je kliknutí na tlačítko, zadání textu do vyhledávacího pole, přidání položky do online nákupního košíku nebo otevření mobilní navigace. Metrika určuje, jak rychle prohlížeč zobrazí vizuální zpětnou vazbu na danou akci v dalším vykresleném snímku. Pokud odezva stránky trvá déle, může uživatel získat dojem, že stránka nefunguje správně a není si jistý, zda se požadovaná akce skutečně vykonala. To může vést k opakovanému klikání na tlačítko nebo zadávání příkazu (Wagner, 2024).

Skóre INP je hodnoceno tak, že výsledek do 200 ms je považován za dobrý, mezi 200 ms a 500 ms vyžaduje zlepšení a hodnota nad 500 ms je již označována jako špatná (Wagner, 2024).

INP je nová metrika rychlosti webu, která v březnu 2024 nahradila předchozí metriku First Input Delay (FID). INP na rozdíl od FID měří celý proces reakce na akci, nejen vstupní zpoždění. FID byla méně náročná a splňovalo ji 95 % webových stránek, což ji činilo nepříliš efektivním ukazatelem. INP je přísnější a ukazuje, že mnoho webů, zejména aplikace využívající JavaScript ve velkém rozsahu, jako jsou React nebo Next.js, vyžaduje optimalizaci (Michálek, 2024).

2.3.1 Hlavní příčiny vysoké hodnoty INP

Mezi hlavní příčiny vysoké hodnoty INP patří:

- **Příliš dlouhé zpracování událostí:** Pokud událost, jako je kliknutí na tlačítko, spustí kód, který zabere hodně času, blokuje to vykreslování stránky. To může nastat, když je v obslužné rutině události zahrnuto příliš mnoho synchronní práce, například výpočty nebo složité operace.
- **Náročné úlohy v JavaScriptu:** Dlouho běžící JavaScriptové úlohy na hlavním vlákně mohou způsobit zpoždění reakce na uživatelskou interakci. Typickým problémem je neefektivní kód v knihovnách, jako je React nebo Angular, který vede k nadměrnému překreslování stránky.
- **Velké balíčky JavaScriptu při načítání stránky:** Při načítání webu mohou velké soubory JavaScriptu prodlužovat dobu, než je stránka schopna reagovat na první vstup uživatele. Problém často souvisí s nadbytečnými knihovnami nebo kódem, který není pro konkrétní stránku vůbec potřeba.
- **Komplexní struktura DOM:** Pokud je HTML stránka složitá, s mnoha vnořenými prvky nebo nadměrnými atributy, trvá delší dobu, než prohlížeč zpracuje změny a vykreslí odpověď na uživatelský vstup (Griffin, 2024).

2.3.2 Optimalizační techniky pro snížení INP

Pro snížení hodnoty INP lze využít následující optimalizační postupy:

- **Odstranění nepoužitého JavaScriptu:** Snížení zátěže na hlavní vlákno odstraněním nevyužívaného kódu. Techniky jako *tree shaking* nebo dělení kódu (*code splitting*) umožňují oddělit potřebný JavaScript od nadbytečného, čímž se urychlí načítání a zpracování.
- **Oddálení načtení méně důležitých skriptů:** Skripty třetích stran, jako jsou chatovací widgety nebo analytické nástroje, je dobré načítat až po zobrazení kritického obsahu stránky.
- **Optimalizování skriptů s vysokou zátěží:** Je třeba identifikovat JavaScriptové soubory, které vyžadují velké množství zdrojů.
- **Rozdělení dlouhých úloh:** Dlouhé úlohy blokují hlavní vlákno a prodlužují dobu odezvy stránky. Rozdělením úloh na menší části pomocí funkce jako *setTimeout* je prohlížeči umožněno aktualizovat obsah a reagovat rychleji.
- **Poskytnutí okamžité zpětné vazby uživateli:** Pokud uživatel provede akci, například klikne na tlačítko, je dobré okamžitě zobrazit vizuální indikaci (např. „Zpracovává se...“) ještě předtím, než je akce potvrzena serverem. Tím se minimalizuje dojem zpožděné odezvy.
- **Zjednodušení DOM struktury:** Komplexní a velký DOM zvyšuje čas potřebný na vykreslení. Aby byla stránka snadno a rychle zpracovatelná, je třeba snížit počet prvků a vnoření (Karel, 2024).

3 Doplňkové metriky a ukazatele výkonu

Vedle metrik Core Web Vitals existují i další ukazatele, které mohou pomoci detailněji analyzovat a optimalizovat výkon webových stránek. S jejich pomocí lze získat širší přehled o procesech spojených s načítáním, zpracováním a interakcemi na webu. Přestože nejsou považovány za hlavní ukazatele výkonu, jejich pochopení a vhodné využití může významně přispět ke zlepšení uživatelského zážitku a celkové technické výkonnosti stránek.

3.1 Time to First Byte

Time to First Byte (TTFB) měří dobu, kterou server potřebuje k odpovědi na požadavek uživatele. TTFB zahrnuje tři hlavní fáze: dobu potřebnou k přenosu požadavku na server, čas na jeho zpracování a dobu k odeslání prvního bytu dat zpět uživateli. Metrika je důležitá pro určení efektivity serverové infrastruktury a rychlosti doručování obsahu (Wagner, Pollard, 2024).

Dobré hodnoty TTFB Google definuje jako dobu do 800 ms, zatímco hodnoty nad 1800 ms jsou považovány za nevyhovující. Protože TTFB nespadá do metrik Core Web Vitals, není nutné brát v úvahu jeho hodnoty jako u ukazatelů LCP, CLS a INP (Wagner, Pollard, 2024).

Mezi příčiny dlouhého TTFB patří zdlouhavé backendové operace, síťová latence nebo absence cache. Optimalizace zahrnuje výběr kvalitního hostingu, nasazení Content Delivery Network (CDN), zjednodušení serverových procesů, nastavení cache a použití moderních protokolů, jako je HTTP/2, pro efektivnější přenos dat (Wagner, Pollard, 2024).

3.2 DOM Content Loaded

Metrika DOM Content Loaded (DCL) označuje okamžik, kdy byl hlavní HTML dokument webové stránky stažen a zpracován. Při dosažení metriky není nutné čekat na načtení dalších zdrojů včetně obrázků, externích stylů, skriptů nebo vložených rámů. Událost nastává během procesu vykreslování stránky a časově se nachází v blízkosti metrik, jako jsou First Contentful Paint (FCP) a First Meaningful Paint (FMP).

DCL je technickou metrikou, která je užitečná zejména při analýze a optimalizaci technické efektivity načítání webu. Sama o sobě však neposkytuje dostatečný vhled do celkové uživatelské zkušenosti, protože se zaměřuje pouze na rychlost zpracování HTML dokumentu (Michálek, 2019).

Pro sledování DCL lze využít nástroje dostupné v prohlížečích, například záložku *Network* ve vývojářských nástrojích Chrome nebo Firefox, kde je možné přesně identifikovat časový bod události (Michálek, 2019).

3.3 First Contentful Paint

Metrika First Contentful Paint (FCP) měří čas od okamžiku, kdy uživatel poprvé přejde na stránku, až po zobrazení prvního textu nebo obrázku na obrazovce. Obsah zahrnuje i prvky vykreslené prostřednictvím CSS vlastnosti *background*, SVG grafiku nebo prvky vytvořené pomocí canvas. FCP je důležitá z hlediska toho, že určuje, kdy uživatel dostává první vizuální signál, že stránka má obsah a začíná se načítat (Walton, 2023).

Pro dosažení dobrého uživatelského dojmu by hodnota FCP neměla překročit 1,8 sekundy. Hodnoty mezi 1,8 a 3 sekundami naznačují potřebu optimalizace a časy nad 3 sekundy jsou považovány za nevyhovující (Walton, 2023).

Optimalizace FCP zahrnuje především minifikaci CSS a JavaScriptu, odstranění zbytečného kódu a zlepšení odezvy serveru, což úzce souvisí s optimalizací metriky TTFB. K dalším krokům patří minimalizace přesměrování, která mohou zbytečně prodlužovat dobu načítání (Walton, 2023).

3.4 Total Blocking Time

Total Blocking Time (TBT) udává časový úsek mezi událostmi First Contentful Paint (FCP) a Time to Interactive (TTI). Je důležité poznamenat, že od února 2023 se metrika TTI již nepoužívá. TBT měří dobu, kdy hlavní vlákno prohlížeče nemohlo reagovat na vstupy uživatele z důvodu provádění dlouhých úloh JavaScriptu. Dlouhá úloha je definována jako úkol běžící v hlavním vlákně déle než 50 ms. Uživatel může tuto situaci vnímat například jako nefunkční kliknutí, protože prohlížeč stále dokončuje zpracování kódu. (Michálek, 2021).

TBT se zaměřuje na problém, kdy stránka ještě není schopná reagovat na vstupy uživatele. Pro dosažení dobrých výsledků by hodnota TBT neměla překročit 200 ms. Hodnoty mezi 200 a 600 ms vyžadují optimalizaci a nad 600 ms již značí závažný problém s výkonem (Michálek, 2021).

Optimalizace TBT spočívá především v úpravě JavaScriptu. Důležité je minimalizovat velikost kódu, identifikovat dlouhé úlohy pomocí vývojářských nástrojů (DevTools, záložka Performance) a optimalizovat konkrétní části kódu. Zároveň je vhodné omezit používání nadbytečných knihoven a minimalizovat závislosti na kódu třetích stran (Michálek, 2021).

3.5 Speed Index

Speed Index (SI) je metrika, která hodnotí rychlost úplného vykreslení a zobrazení obsahu stránky uživateli. Na rozdíl od metrik jako First Contentful Paint (FCP) nebo Largest Contentful Paint (LCP), které hodnotí konkrétní okamžiky načítání obsahu, SI sleduje celý proces vizuálního zaplnění stránky. Metrika tedy ukazuje, jak rychle se uživatelům zobrazí plný obsah stránky na jejich obrazovce (Keeton, 2023).

Hodnota Speed Indexu se měří v sekundách. Pro dobrý uživatelský zážitek by SI neměl přesáhnout 3,4 sekundy. Hodnoty mezi 3,4 a 5,8 sekundami naznačují potřebu optimalizace, zatímco hodnoty nad 5,8 sekundy jsou považovány za špatné (Keeton, 2023).

Optimalizace SI zahrnuje několik kroků. Prvním z nich je snížení doby provádění JavaScriptu, jelikož dlouhé úlohy blokují vykreslování viditelného obsahu. Důležité je také odložit některé skripty až po vykreslení klíčových prvků na stránce. Minimalizace použití zbytečného JavaScriptu a redukce kódu třetích stran přispívá k lepšímu výkonu webu. Užitečné je rovněž minifikovat CSS a HTML, čímž se snižuje zátěž hlavního vlákna při zpracování (Keeton, 2023).

3.6 Lighthouse Performance Score

Lighthouse Performance Score (LPS) je syntetická metrika, která slučuje všechny aktuálně důležité metriky rychlosti načítání a výkonu webu do jednoho výsledného skóre. Skóre je reprezentováno číslem na škále od 0 do 100, kde vyšší hodnota značí lepší výkon. LPS lze považovat za hlavní syntetickou metriku, protože vychází z laboratorních měření prováděných strojem, nikoliv z dat reálných uživatelů. Každá metrika v LPS má určitou váhu, která ovlivňuje její vliv na celkový výsledek (Michálek, 2021).

Tab. 1: Váhy a ideální hodnoty metrik tvořících Lighthouse Performance Score

Metrika	Váha	Ideální hodnota
First Contentful Paint (FCP)	10 %	≤ 1,8 s
Largest Contentful Paint (LCP)	25 %	≤ 2,5 s
Speed Index (SI)	10 %	≤ 3,4 s
Total Blocking Time (TBT)	30 %	≤ 0,2 s
Cumulative Layout Shift (CLS)	25 %	≤ 0,1 s

Zdroj: vlastní zpracování dle Michálek (2021)

Od verze Lighthouse 10 jsou všechny uvedené metriky zahrnuty do výpočtu LPS a tvoří základ pro hodnocení rychlosti a výkonu webových stránek (Michálek, 2021).

LPS nabízí tři základní úrovně hodnocení, které určují kvalitu výkonu webu:

- **Zelená (dobrá rychlost):** skóre mezi 90 a 100.
- **Oranžová (vyžaduje zlepšení):** skóre mezi 50 a 89.
- **Červená (špatná rychlost):** skóre mezi 0 a 49.

I když LPS poskytuje jednoduché a srozumitelné hodnocení, jeho interpretace by měla být vždy doplněna kontextem. Například není vhodné srovnávat LPS mezi fundamentálně odlišnými typy webů, jako jsou jednoduché prezentační stránky, komplexní e-shopy nebo zpravodajské portály, které mohou mít odlišné požadavky a technická omezení (Michálek, 2021).

Nástroje jako PageSpeed Insights nebo Chrome DevTools umožňují najít oblasti, které je třeba zlepšit. Přesto není vždy nezbytné usilovat o dosažení maximálního skóre 100 bodů, protože taková snaha může být v některých případech neefektivní a nákladná (Michálek, 2021).

4 Nástroje pro měření výkonu

K posuzování výkonu webových stránek a aplikací slouží řada nástrojů, které vyhodnocují jak syntetické metriky, tak data sbíraná od reálných uživatelů (RUM). Kapitola představuje přehled pěti oblíbených nástrojů: PageSpeed Insights, Google Lighthouse, GTmetrix, WebPageTest a Pingdom Tools. Každý z těchto nástrojů přináší specifický pohled a nabízí jedinečné metody analýzy.

4.1 PageSpeed Insights

PageSpeed Insights (PSI) je analytický nástroj od společnosti Google, který zdarma poskytuje informace o rychlosti a výkonu webových stránek. Na trhu je k dispozici již několik let a prochází pravidelnými aktualizacemi. Díky své jednoduchosti je vhodný nejen pro zkušené uživatele, ale i pro začátečníky (Trauzold, 2020). Michálek (2024), přední český odborník na optimalizaci webů, doporučuje začít právě s PSI a zaměřit se na dosažení skóre alespoň 80 bodů na desktopu i mobilu, přičemž je důležité prioritně řešit červeně zvýrazněné problémy. Dosažení maximálního skóre 100 bodů není nutné.

4.1.1 Analýza výkonu

PSI poskytuje detailní analýzu načítání webu a zároveň navrhuje konkrétní kroky pro zlepšení výkonu. Například může doporučit odstranění zdrojů blokujících vykreslení, optimalizaci obrázků nebo odstranění nevyužívaného JavaScriptu. Každé doporučení je doplněno odhadem času, který lze optimalizací ušetřit (Trauzold, 2020).

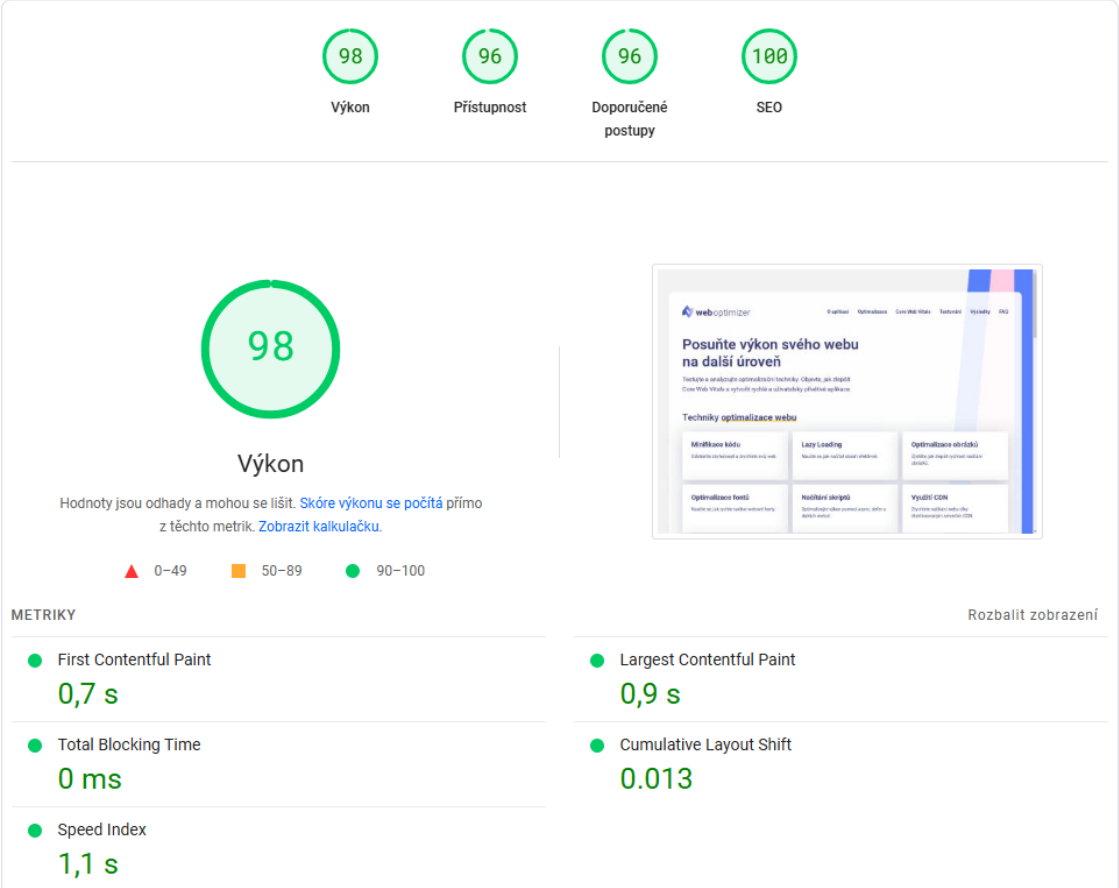
4.1.2 Použití PSI

Použití PSI je velmi intuitivní. Stačí na oficiálním webu nástroje zadat URL adresu webové stránky nebo konkrétní podstránky a spustit analýzu. Výsledky jsou prezentovány samostatně pro mobilní zařízení a desktopy, což reflektuje prioritu přístupu *mobile-first*. Google preferuje optimalizaci pro mobilní zařízení před desktopy, protože většina uživatelů přistupuje na web právě z mobilních zařízení (Trauzold, 2020).

PSI využívá data z Chrome UX Report (CrUX), která zahrnují informace o uživatelských zkušenostech s metrikami jako FCP, INP, LCP, CLS a experimentální TTFB. Data jsou shromažďována vždy za předchozích 28 dní. U méně navštěvovaných webů však CrUX data nejsou k dispozici, protože nedosahují dostatečného množství interakcí od reálných uživatelů (Google for Developers, 2024).

Hodnocení výkonu je vyjádřeno skórem na škále od 0 do 100 bodů, přičemž se rozlišují tři úrovně:

- **90 a více bodů**: Výborný výkon.
- **50 až 89 bodů**: Oblast vyžadující zlepšení.
- **49 a méně bodů**: Nevyhovující výkon (Google for Developers, 2024).



Obr. 1: Výsledky měření v nástroji PageSpeed Insights
Zdroj: vlastní zpracování (2025)

4.2 Google Lighthouse

Google Lighthouse je bezplatný open-source nástroj od společnosti Google sloužící k analýze kvality webových stránek. Na rozdíl od PageSpeed Insights (PSI), který kombinuje syntetická data s reálnými uživatelskými daty z Chrome UX Reportu (CrUX), Lighthouse provádí výhradně syntetické testy (Pol, 2023). Výsledky testů často závisí na výkonu zařízení použitého pro testování, což může vést k odlišným hodnotám oproti podmínkám běžných uživatelů, zejména u vývojářů s výkonnými počítači (Michálek, 2021).

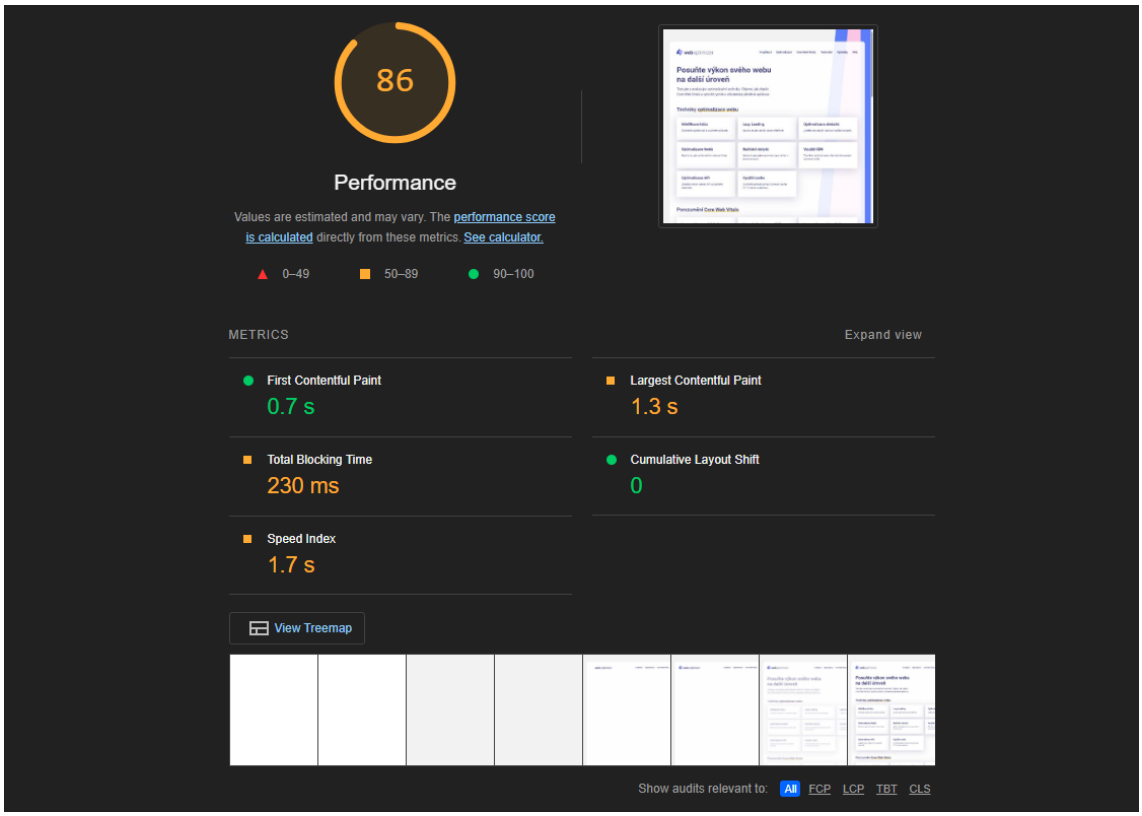
4.2.1 Analýza výkonu

Lighthouse se zaměřuje na čtyři hlavní oblasti analýzy: výkon (Performance), přístupnost (Accessibility), dodržování osvědčených postupů (Best Practices) a základy SEO. Poskytuje ucelený pohled na technickou kvalitu stránek a pomáhá identifikovat možnosti pro zlepšení (Pol, 2023).

4.2.2 Výhody Lighthouse

Jednou z hlavních výhod je snadná dostupnost přímo v Google Chrome DevTools. Lighthouse je možné spustit prostřednictvím stejnojmenné záložky v nástroji. Uživatelé mohou vybírat mezi

testováním na mobilních zařízeních nebo desktopu. Výsledky zahrnují skóre pro každou analyzovanou oblast a konkrétní doporučení pro optimalizaci (Michálek, 2021).



Obr. 2: Výsledky měření z nástroje Lighthouse
Zdroj: vlastní zpracování (2025)

4.2.3 Omezení a doporučení

Přestože Lighthouse nenahrazuje pokročilé nástroje detailní analýzy nebo dlouhodobé sledování výkonu, je ideální pro pravidelnou kontrolu a základní optimalizaci webů. Umožňuje odhalit hlavní problémy a navrhnout řešení pro zlepšení technické kvality stránek i jejich pozici ve vyhledávačích (Michálek, 2021).

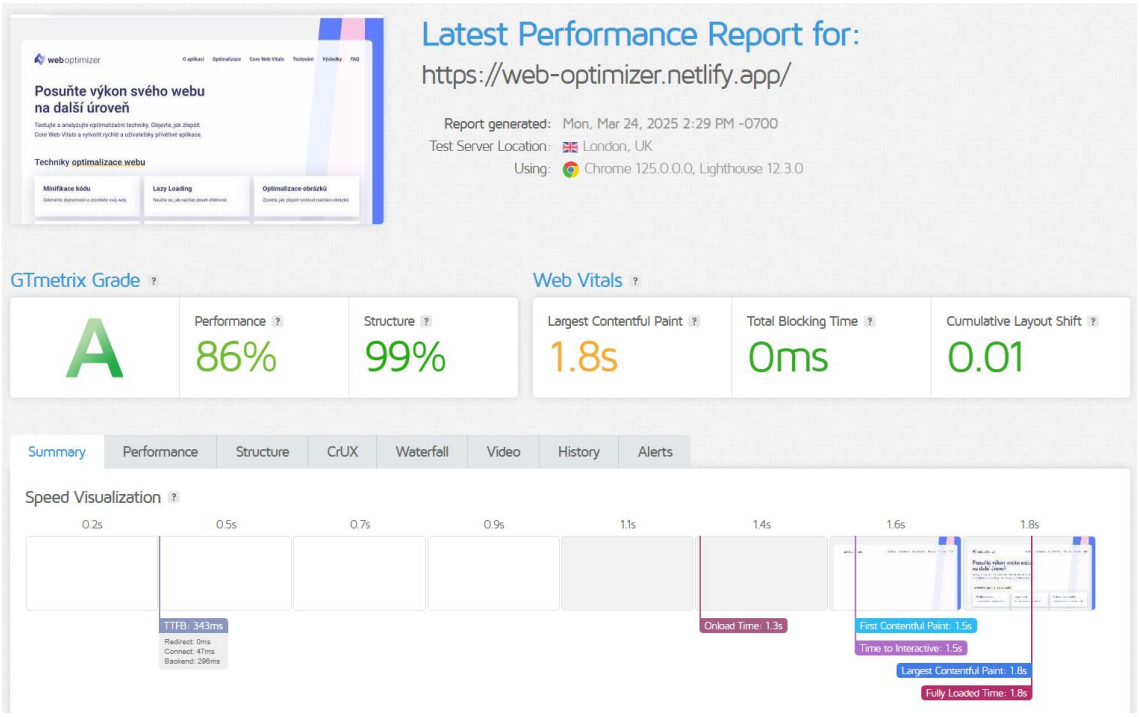
4.3 GTmetrix

GTmetrix je jedním z nejpopulárnějších nástrojů analýzy výkonu webových stránek. Umožňuje uživatelům získat detailní přehled o rychlosti načítání a poskytuje konkrétní doporučení pro optimalizaci. Kombinací výsledků z Google PageSpeed Insights a YSlow nástroj pokrývá širokou škálu metrik a eliminuje nutnost využívat další platformy pro testování (Alexandrea, 2023).

4.3.1 Použití GTmetrix

Při použití GTmetrix stačí zadat URL webové stránky a spustit analýzu. Bezplatná verze zahrnuje výběr testovacích serverů v sedmi lokalitách, jako jsou Vancouver, Dallas, Londýn, Hongkong, Mumbai, São Paulo a Sydney. Testování probíhá na desktopové verzi prohlížeče Chrome

s možností simulace různých typů připojení, včetně neomezeného připojení nebo pomalého 3G. Pro přesnější výsledky je doporučeno test opakovat v různých časech a zohlednit cílové publikum výběrem vhodného testovacího místa (Alexandrea, 2023).



Obr. 3: Výsledky měření v nástroji GTmetrix
Zdroj: vlastní zpracování (2025)

4.3.2 Srovnání s jinými nástroji

Při porovnání s dalšími nástroji, jako jsou Pingdom a WebPageTest, má GTmetrix odlišnost v určitých ohledech. Každý z těchto nástrojů používá vlastní sadu doporučení a testovacích metodologií, což pochopitelně vede k rozdílům ve výsledcích. GTmetrix kombinuje 27 doporučení z Google PageSpeed a 19 z YSlow, zatímco například WebPageTest analyzuje webové stránky podle šesti vlastních doporučení. Dále se liší v počtu a umístění testovacích serverů. GTmetrix nabízí sedm lokalit s celkem 28 dedikovanými servery, to umožňuje uživatelům vybrat si místo nejblíže jejich cílovému publiku pro přesnější měření. Navíc GTmetrix ve výchozím nastavení měří čas plného načtení stránky. Interval zahrnuje dobu od zahájení načítání až po okamžik, kdy po dvě sekundy nedochází k žádné síťové aktivitě (Morey, 2021).

4.4 WebPageTest

WebPageTest je výkonný open-source nástroj. Spuštěn byl v roce 2008 a stal se jedním z nejstarších nástrojů pro testování výkonu webových stránek. Nejprve byl interním testovacím nástrojem v rámci společnosti AOL, což dokazuje jeho dlouhou historii a vývoj. V roce 2011 byl WebPageTest uvolněn jako open-source projekt, to umožnilo jeho široké využití a přístupnost pro vývojáře a správce webových stránek (Holcombe, 2024).

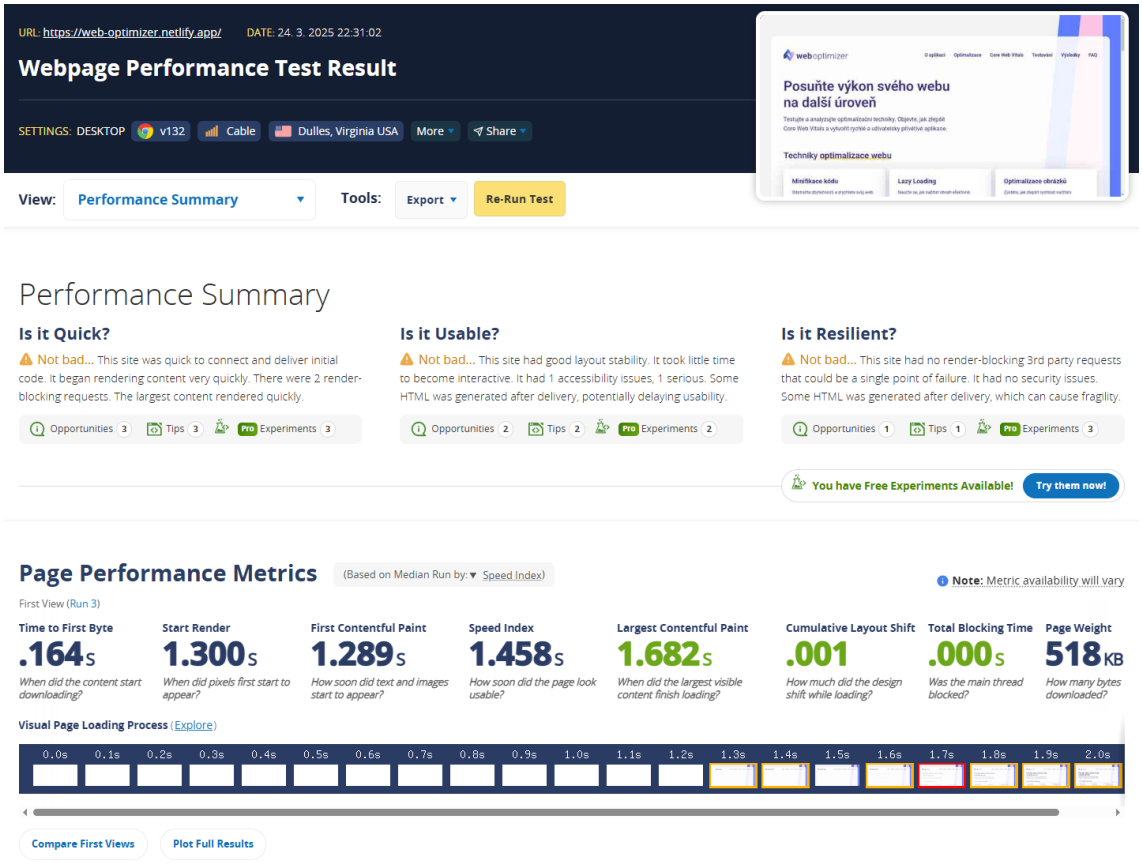
4.4.1 Klíčové funkce

WebPageTest poskytuje podrobnou analýzu výkonu, která zahrnuje:

- **Měření doby načítání**: Umožňuje sledovat čas potřebný k načtení stránky a jednotlivých prvků.
- **Simulace testů**: Testy lze provádět z různých geografických lokalit, což pomáhá pochopit, jak se stránka načítá uživatelům na různých místech.
- **Podpora různých prohlížečů**: Nástroj je kompatibilní s hlavními prohlížeči jako Chrome, Firefox a Safari.
- **Pokročilé metriky**: Mezi metriky patří Speed Index, Time to First Byte (TTFB) a First Contentful Paint (FCP), které poskytují detailní pohled na výkon stránky (Holcombe, 2024).

4.4.2 Použití WebPageTest

Použití WebPageTest je považováno za jednoduché. Po registraci bezplatného účtu je zadána URL adresa stránky k otestování a vybrána konfigurace testu. Pro dosažení nejlepších výsledků se doporučuje provádět testy na mobilních i desktopových zařízeních a volit testovací lokalitu co nejblíže serveru webu. Po dokončení testu je vygenerována podrobná zpráva o výkonu stránky obsahující metriky, jako jsou TTFB, Start Render a Speed Index (Holcombe, 2024).



Obr. 4: Výsledky měření v nástroji WebPageTest
Zdroj: vlastní zpracování (2025)

4.4.3 Výhody pravidelného používání

WebPageTest je obzvláště užitečný pro pravidelné sledování výkonu webových stránek, zejména při větších změnách, jako jsou redesigny nebo migrace na nového poskytovatele hostingu. Díky historickým datům lze sledovat trendy výkonu a včas identifikovat potenciální problémy, což přispívá k lepší uživatelské zkušenosti a vyšším pozicím ve vyhledávačích (Holcombe, 2024).

4.4.4 Srovnání s jinými nástroji

Na rozdíl od jiných nástrojů pro testování výkonu, jako jsou Google PageSpeed Insights nebo GTmetrix, WebPageTest nabízí hlubší analýzu a flexibilitu při testování. Zatímco Google PageSpeed Insights se zaměřuje na skóre a doporučení pro optimalizaci, WebPageTest poskytuje detailní pohled na skutečné načítání stránek v různých podmínkách (Holcombe, 2024).

4.5 Pingdom Tools

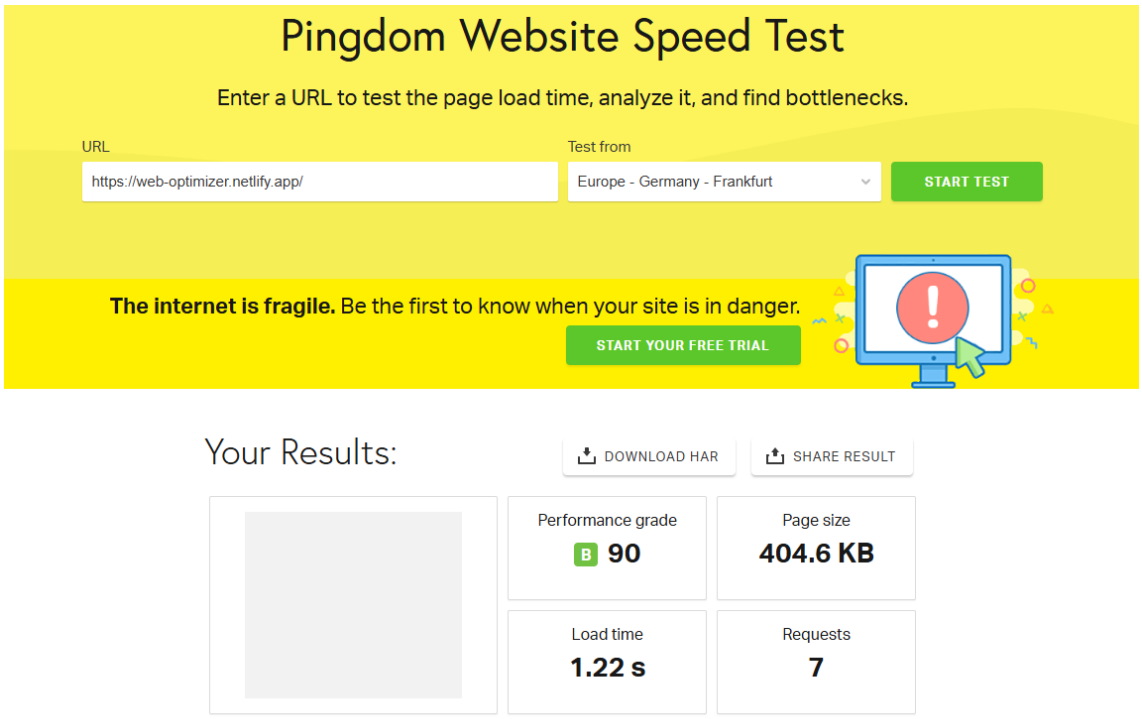
Další nástroj Pingdom Tools byl vyvinut švédskou společností Pingdom, nyní patřící pod SolarWinds. Nástroj se specializuje na sledování výkonu webových stránek a nabízí uživatelům možnost testovat načítání stránek z různých geografických lokalit po celém světě (Jackson, 2024).

4.5.1 Geografické lokality

Testování z různých lokalit je zásadní pro získání přesného obrazu o výkonu webových stránek, protože umožňuje zohlednit například síťovou latenci, geografickou vzdálenost mezi serverem a uživateli. Pingdom Tools poskytuje uživatelům možnost provádět testy rychlosti z několika lokalit na různých kontinentech. Mezi dostupné testovací lokality patří Tokio, Frankfurt, Londýn, Washington D.C., San Francisco, Sydney a São Paulo (Jackson, 2024).

4.5.2 Použití Pingdom Tools

Použití Pingdom Tools je intuitivní. Uživatelé jednoduše zadávají URL adresu webové stránky a vybírají lokalitu pro test. Nástroj následně generuje podrobnou zprávu o výkonu, která zahrnuje celkový čas načítání, velikost stránky a počet požadavků. Klíčovým prvkem analýzy je *waterfall chart*, který vizualizuje načítání jednotlivých komponentů stránky a pomáhá identifikovat potenciální problémy s výkonem (Jackson, 2024).



Obr. 5: Výsledky měření v nástroji Pingdom Tools
Zdroj: vlastní zpracování (2025)

4.5.3 Srovnání s jinými nástroji

Pingdom Tools se odlišuje od ostatních nástrojů pro testování výkonu webových stránek svým zaměřením na reálné uživatelské monitorování. Od služeb GTmetrix a WebPagetest se liší především metodou měření času načítání. Každý z těchto nástrojů poskytuje různé výsledky v závislosti na testovací lokalitě, která hraje důležitou roli v určení latence a kvality síťového připojení. Například, pokud je web hostován v San Franciscu, testování z místa jako Stockholm může vykazovat delší dobu načítání než testování z Vancouveru. Pingdom Tools měří pouze *onload time*, což znamená, že se zaměřuje na okamžik, kdy je stránka považována za plně načtenou, zatímco jiné nástroje jako GTmetrix a WebPagetest nabízejí možnost měřit *fully loaded time*, to zahrnuje i dodatečné prvky stránky. Rozdíly v metodologii a testovacích lokalitách vysvětlují variabilitu výsledků mezi jednotlivými nástroji (Morey, 2021).

4.6 Ostatní nástroje

Kromě hlavních nástrojů pro měření výkonu, jako jsou PageSpeed Insights, Google Lighthouse, GTmetrix, WebPageTest a Pingdom Tools, existuje řada dalších užitečných nástrojů, které mohou pomoci při analýze a optimalizaci webových stránek.

4.6.1 PageSpeed.cz Tester

PageSpeed.cz je nástroj postavený na Google Lighthouse a datech z Chrome UX Reportu, jenž umožňuje testovat rychlost webových stránek. Nabízí také placenou verzi pro profesionály s označením PLUS, nabízející pokročilé funkce a detailnější analýzy (Michálek, 2016).

4.6.2 Google Analytics

Google Analytics je široce používaný nástroj pro sledování návštěvnosti a chování uživatelů na webových stránkách. Pro vývojáře je obzvláště užitečný při rozšíření o nástroje jako Trackomatic a Technical Performance Dashboard, které umožňují detailní přehled o technických parametrech webu, včetně rychlosti načítání v různých prohlížečích a regionech (Michálek, 2016).

4.6.3 Chrome DevTools

Chrome DevTools je sada vývojářských nástrojů integrovaná v prohlížeči Google Chrome. Například v záložce *Network* lze sledovat načítání jednotlivých zdrojů, omezit rychlost připojení pro simulaci různých podmínek nebo identifikovat události jako DOMContentLoaded a Load (Michálek, 2016).

4.6.4 SpeedCurve

SpeedCurve je nástroj pro kontinuální monitorování rychlosti webových stránek, který poskytuje podrobné reporty sloužící jako podklad pro optimalizaci výkonu. Kombinuje syntetické testování v kontrolovaném prostředí s reálnými uživatelskými daty, což umožňuje komplexní analýzu v porovnání s konkurencí. Mezi jeho klíčové funkce patří nastavení výkonnostních limitů, sledování metrik jako Core Web Vitals a integrace s CI/CD procesy (Everts, 2023).

5 Optimalizační techniky

V dnešní digitální době rychlost načítání webových stránek ovlivňuje nejen uživatelskou zkušenost, ale také pozici ve výsledcích vyhledávání. Rychlé načítání může ovlivnit rozhodování uživatelů při nákupu a jejich celkovou spokojenost s webem. Kapitola o optimalizačních technikách se zaměřuje na různé metody, které mohou pomoci dosáhnout rychlejšího načítání, snížit potřebu zdrojů a optimalizovat celkový výkon webových aplikací. V následujících podkapitolách jsou rozebrány jednotlivé techniky, jako je minifikace kódu, lazy loading, komprese obrázků, optimalizace API volání a využití Content Delivery Network (CDN), spolu s jejich praktickými přínosy.

5.1 Minifikace kódu

Minifikace kódu představuje proces odstranění nadbytečných znaků z kódu včetně mezer, komentářů a nových řádků při zachování jeho plné funkčnosti. Nejčastěji se aplikuje na soubory jako JavaScript, CSS a v některých případech i HTML. Hlavním cílem je zmenšení velikosti souborů, což vede k rychlejšímu načítání webových stránek a efektivnějšímu přenosu dat mezi serverem a klientem (Sharma, 2023).

5.1.1 Nástroje pro minifikaci

V dnešní době se minifikace kódu provádí téměř výhradně pomocí automatizovaných build nástrojů, které nahradily starší *task runners* jako Grunt a Gulp. Mezi nejpopulárnější nástroje současnosti patří Webpack, Vite, Rollup, Esbuild a SWC. Nabízejí komplexní řešení optimalizace kódu a zahrnují funkce jako minifikace, komprese a optimalizace souborů. Navíc oproti starším nástrojům jsou výrazně rychlejší. Níže jsou uvedeny nejvýraznější nástroje:

- **Vite:** Moderní buildovací nástroj poskytující rychlý vývoj. Využívá ES moduly a efektivní Hot Module Replacement (HMR), což výrazně urychluje vývojový proces. V produkčním módu automaticky využívá minifikaci prostřednictvím nástrojů jako Esbuild. Je ideální pro webové frameworky jako React nebo Vue (Haworth, 2022).
- **Webpack:** Pravděpodobně nejznámější modulární bundlovací nástroj, který nabízí široké možnosti konfigurace a optimalizace. Je vhodný pro velké projekty s mnoha závislostmi a aktivy, ale jeho konfigurace může být složitá a vyžaduje hlubší znalosti (Lazaris, 2024).
- **Rollup:** Zaměřuje se na vytváření optimalizovaných balíčků s minimálním přetěžováním. Je oblíbený především při tvorbě knihoven díky podpoře *tree-shaking*, což minimalizuje zahrnutí nepotřebného kódu (Lazaris, 2024).
- **Esbuild:** Extrémně rychlý bundlovací nástroj, napsaný v jazyce Go. Podporuje moderní JavaScript i TypeScript a jeho rychlost ho činí ideálním pro malé až střední projekty (Haworth, 2022).
- **SWC (Speedy Web Compiler):** Rychlý kompilátor založený na Rustu, který se používá pro kompilaci a minifikaci JavaScriptu a TypeScriptu. Je kompatibilní s Webpackem a Jestem, v současnosti bohužel nepodporuje bundlování (Lazaris, 2024).

5.2 Lazy loading

Lazy loading je technika, která odkládá načítání prvků stránky, jako jsou obrázky, videa nebo iframy, dokud nejsou potřeba. Například, když se uživatel přiblíží k danému obsahu při scrollování. Sníží se počáteční zátěž serveru, zkrátí se doba načítání stránky a optimalizuje se využití šířky pásma. Moderní prohlížeče nabízejí nativní podporu této techniky prostřednictvím atributu *loading="lazy"*, který lze snadno implementovat v HTML. Pro složitější situace, například při dynamickém načítání obsahu, lze využít JavaScript a Intersection Observer API, které poskytují větší flexibilitu a kontrolu nad procesem načítání (Michálek, 2019).

5.3 Komprese obrázků

Obrázky často tvoří většinu objemu dat webové stránky. Komprese je proces, který snižuje velikost souborů obrázků, aby se urychlil jejich přenos a načítání. Existují dva hlavní typy komprese: ztrátová a bezztrátová.

5.3.1 Ztrátová komprese

Ztrátová komprese odstraňuje část dat z obrázku, což snižuje jeho velikost, nicméně může mírně snížit kvalitu obrazu. Hodí se zejména pro fotografie a obrázky s mnoha barvami, kde je kladen důraz na rychlost načítání před dokonalou kvalitou. Mezi nejčastěji používané formáty využívající ztrátovou kompresi patří JPEG a WebP, které díky svým vlastnostem výrazně snižují velikost souborů při zachování vizuálně přijatelné kvality (Rossi, 2024).

5.3.2 Bezztrátová komprese

Bezztrátová komprese umožňuje zmenšit velikost obrázků bez ztráty jakýchkoli detailů nebo kvality. Nejvíce se využívá u grafiky, log nebo obrázků s textem a průhledností, kde je důležité zachovat ostré hrany a přesné barvy. Formáty jako PNG a GIF jsou typickými příklady, protože dokážou uchovat všechny informace v původní kvalitě, přičemž jsou vhodné pro obrázky s omezeným počtem barev nebo s požadavkem na transparentnost (Rossi, 2024).

5.3.3 Nástroje pro kompresi obrázků

Pro snížení velikosti obrázků a optimalizaci jejich načítání je k dispozici několik užitečných nástrojů a služeb. Například se jedná o:

- **TinyPNG**: Online nástroj specializující se na kompresi PNG a JPEG souborů s minimální ztrátou kvality.
- **ImageOptim**: Aplikace pro macOS, která optimalizuje obrázky odstraněním nepotřebných dat a kompresí bez ztráty kvality.
- **Squoosh**: Webová aplikace od společnosti Google umožňující kompresi obrázků s náhledem změn v reálném čase a podporou různých formátů.
- **ShortPixel**: Plugin pro WordPress, který automaticky optimalizuje obrázky při jejich nahrávání na web, podporující jak ztrátovou, tak bezeztrátovou kompresi (Rossi, 2024).

5.3.4 Moderní formáty obrázků

Pro moderní webové aplikace je výhodné používat nové formáty obrázků, které nabízejí lepší kompresní poměr a kvalitu než starší formáty jako JPEG, PNG nebo GIF.

- **WebP**: Formát vyvinutý společností Google, nabízející lepší kompresní poměr než JPEG a PNG, a je podporován většinou moderních prohlížečů.
- **AVIF**: Novější formát založený na technologii AV1, který poskytuje ještě lepší kompresní poměry než WebP. AVIF podporuje ztrátovou i bezeztrátovou kompresi, průhlednost a vysoký dynamický rozsah (HDR). Přestože podpora v prohlížečích stále roste, je doporučeno poskytovat záložní formáty pro starší prohlížeče.
- **SVG**: Scalable Vector Graphics (SVG) je formát pro vektorovou grafiku, ideální pro loga, ikony a ilustrace. SVG umožňuje neomezené škálování bez ztráty kvality a často má velmi malou velikost souboru. Díky textové povaze je také snadno upravitelný a stylovatelný pomocí CSS (Michálek, 2020).

5.4 Optimalizace API volání

API (Application Programming Interface) je rozhraní, které umožňuje vzájemnou komunikaci mezi různými softwarovými aplikacemi. API definuje pravidla a formáty, podle kterých si aplikace mohou vyměňovat data nebo spouštět specifické funkce. Díky API je například možné, aby webové aplikace získávaly data z databází, komunikovaly se servery nebo integrovaly služby třetích stran.

Rychlost a spolehlivost API hrají zásadní roli v plynulém chodu webových aplikací. Důležité je sledovat dobu odezvy, latenci, dostupnost a využití CPU, protože právě zmíněné metriky ovlivňují celkový výkon. Pro zlepšení fungování API se doporučuje například stránkování pro rozdělení velkých dat, což snižuje zátěž serveru. Asynchronní zpracování požadavků zase pomáhá obsloužit více klientů současně. Cachování často používaných dat a optimalizace databázových dotazů přinášejí další zrychlení, zatímco komprese dat snižuje objem přenášených informací a zkracuje rychlost odezvy. Pravidelný monitoring výkonu pomáhá odhalit slabá místa a včas řešit případné problémy (Tarugu, 2023).

5.5 Využití CDN

Content Delivery Network (CDN) je síť serverů rozmístěných po celém světě, která zajišťuje rychlé doručování webového obsahu uživatelům. Funguje tak, že ukládá statické soubory, jako jsou obrázky, JavaScript nebo CSS, na servery v různých lokalitách, aby mohly být uživateli doručeny z nejbližšího možného místa. Díky tomuto přístupu snižuje latenci, zvyšuje rychlost načítání a zlepšuje uživatelskou zkušenost (Cloudfare, 2024).

5.5.1 Výhody využívání CDN

Mezi hlavní výhody využívání CDN patří:

- **Zvýšení rychlosti načítání:** Doručováním obsahu z geograficky blízkých serverů se zkracuje doba načítání stránek, což vede k lepší uživatelské zkušenosti.

- **Snížení zátěže původního serveru:** Distribucí obsahu na více serverů se snižuje zatížení hlavního serveru, což může zlepšit jeho výkon a stabilitu.
- **Zvýšení dostupnosti a spolehlivosti:** Díky redundanci serverů v CDN je obsah dostupný i v případě výpadku některého z nich, což zvyšuje celkovou spolehlivost webu.
- **Ochrana proti DDoS útokům:** CDN může pomoci rozložit a zmírnit dopad distribuovaných útoků odmítnutí služby, čímž chrání web před přetížením (Webglobe, 2024).

5.5.2 Nevýhody využívání CDN

Naopak mezi nevýhody sítě CDN patří zejména:

- **Výpadek CDN**: Pokud dojde k výpadku CDN, web se může špatně zobrazit nebo fungovat nesprávně, protože externí soubory jako CSS nebo JavaScript se nenačtou.
- **Ztížený off-line vývoj**: Při vývoji webu na *localhostu* mohou externí soubory z CDN způsobit problémy, pokud není dostupné internetové připojení.
- **Spojení na další doménu**: Načítání souborů z CDN může trvat déle než načítání ze stejné domény, odkud pochází HTML kód, protože se musí vytvořit spojení na novou doménu.
- **Různé verze knihoven**: Používání různých verzí CSS/JS knihoven napříč weby může vést k problémům s kompatibilitou a omezením výhod cachování.
- **Bezpečnostní rizika**: Připojování souborů z cizích CDN může představovat bezpečnostní riziko, protože obsah, nad kterým nemá provozovatel kontrolu, může být upraven s úmyslem napáchat škodu (Jahoda, 2016).

5.5.3 Používání CDN v praxi

CDN se běžně používají v reálných projektech, kde je vyžadována rychlá a spolehlivá distribuce obsahu. Například velké e-commerce platformy nebo globální webové aplikace často využívají CDN pro zajištění co nejlepší uživatelské zkušenosti. Nicméně, pro menší projekty nebo projekty s nízkou návštěvností může být výhodnější stahovat knihovny lokálně, aby se snížila závislost na třetích stranách (Cloudflare, 2024).

6 Použité technologie

V rámci vývoje webové aplikace byly použity technologie pro zajištění spolehlivého fungování, přehledného designu a optimalizaci zdrojů. Následující podkapitoly popisují HTML pro strukturování obsahu, SCSS pro stylování, React knihovnu pro dynamické uživatelské rozhraní a Netlify pro hosting a automatické nasazování. Pro sestavování projektu byl využit nástroj Vite, který je popsán v podkapitole 5.1.1 Nástroje pro minifikaci.

6.1 HTML

HTML (HyperText Markup Language) je značkovací jazyk pro strukturování obsahu webových aplikací. Poskytuje rámec pro organizaci prvků na stránce, jako jsou nadpisy, odstavce nebo multimediální obsah, a podporuje přístupnost prostřednictvím sémantických značek, jako jsou *<header>*, *<main>* nebo *<footer>*. Moderní standard HTML5 zahrnuje nové prvky pro práci s multimédii, formuláři a dalšími interaktivními prvky, což zvyšuje efektivitu vývoje a výkon aplikací (MDN Web Docs, 2024).

6.2 SCSS

SCSS (Sassy CSS) je rozšíření CSS, které přináší pokročilé možnosti, jako jsou proměnné, zanořování, mixiny a dědičnost stylů. Díky zmíněným vlastnostem zjednodušuje a zpřehledňuje správu stylů, což je obzvlášť užitečné u větších projektů. SCSS umožňuje snadno přizpůsobovat design a udržovat kód v čistém a přehledném stavu (Yadav, 2023).

6.3 React

React je JavaScriptová knihovna vyvinutá společností Meta (dříve Facebook) pro tvorbu uživatelských rozhraní. Komponentový přístup Reactu umožňuje opakované použití kódu a zjednodušuje vývoj dynamických a interaktivních aplikací. Mezi hlavní výhody patří rychlé vykreslování a správa stavu pomocí knihovny React Hooks (Copes, 2020).

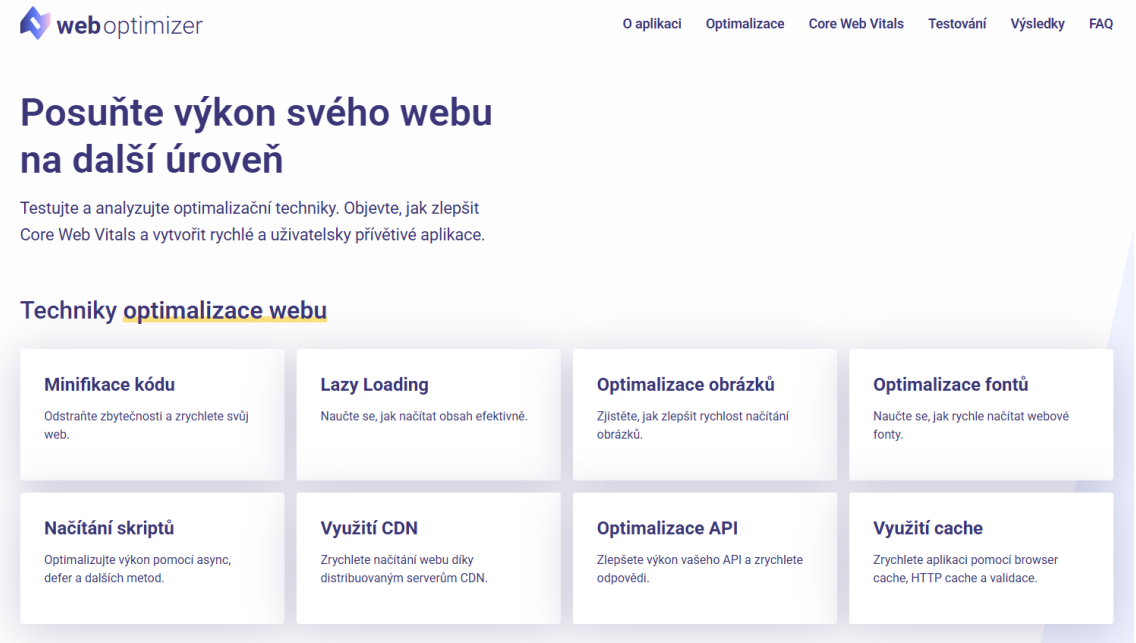
6.4 Netlify

Netlify je platforma pro hosting a automatické nasazování webových aplikací. Umožňuje jednoduchý proces nasazení z Git repozitáře a nabízí funkce jako automatické buildy, správu domén a zabezpečení pomocí HTTPS. Díky podpoře serverless funkcí a dalších nástrojů poskytuje rozměrné řešení pro moderní webový vývoj (Netlify Docs, 2025).

7 Návrh a implementace aplikace

Praktická část bakalářské práce se zaměřuje na vytvoření demonstrační aplikace, která umožňuje testovat a názorně prezentovat jednotlivé techniky optimalizace webových stránek probírané v teoretické části.

Aplikace je pojmenována Web Optimizer a je postavena na technologii React s využitím Vite pro rychlé sestavení a vývoj. Ke stylování je použito SCSS a interaktivita je tvořena pomocí React komponent. Každá optimalizační technika je implementována na samostatné testovací stránce, kde lze přímo porovnat výkon před a po optimalizaci. K analýze výkonu je využívána knihovna web-vitals pro měření v reálném čase a PageSpeed Insights API. Celý projekt je dostupný online na *https://web-optimizer.netlify.app/* a jeho zdrojový kód je uložen v repozitáři na GitHubu na adrese *https://github.com/tomasdevs/web-optimizer*.



Obr. 6: Část úvodní stránky aplikace
Zdroj: vlastní zpracování (2025)

7.1 Architektura aplikace

Web Optimizer je navržen jako Single Page Application (SPA) s více podstránkami, mezi kterými uživatel přepíná pomocí React Routeru. Díky tomu není potřeba opětovné načítání celého webu, jednotlivé stránky se dynamicky renderují na straně klienta.

7.1.1 Datový model

React jako frontendová knihovna není koncipován tak, aby měl pevně definovaný datový model, jako je tomu například u objektově orientovaného programování nebo relačních databází. Namísto toho pracuje s komponentovým modelem, kde každá komponenta může mít svůj vlastní stav (*useState*) nebo například získávat data z API pomocí *useEffect*. V aplikaci se správa dat soustředí na:

- **Lokální stav komponent (useState)** – Používá se pro ukládání dynamických dat, například výsledků testů v komponentě TestPageSpeed.
- **Asynchronní získávání dat (fetch)** – Data o výkonu webových stránek se načítají přes PageSpeed Insights API. Volání API se spouští na základě uživatelské akce, nikoli automaticky při načtení stránky.
- **URL parametry (useSearchParams)** – Testovací stránky umožňují přepínání mezi neoptimalizovanými a optimalizovanými verzemi pomocí URL parametrů (např. *optimized=false* vs. *optimized=true*).
- **Předávání dat mezi komponentami (props)** – Výsledky testů nebo další informace jsou předávány mezi komponentami, aby bylo možné dynamicky měnit obsah aplikace.

7.1.2 Struktura souborů

Pro přehlednost je projekt rozdělen do několika hlavních složek:

- **src/components** – Obsahuje React komponenty, například navigaci, karty a rozvržení stránky.
- **src/pages** – Každá testovací stránka a hlavní části aplikace jako *Home.jsx*, *TestingPage.jsx*, *Optimization.jsx*, *CoreWebVitals.jsx* a další.
- **src/styles** – SCSS soubory pro stylování celé aplikace.
- **src/App.jsx** - Hlavní soubor aplikace s importem všech stránek a jejich výpisem pomocí React Routeru. Definuje navigaci a zajišťuje přesměrování mezi sekcemi aplikace.
- **src/main.jsx** – Vstupní bod aplikace, kde se inicializuje React aplikace a připojuje se k DOMu.
- **public/assets** – Statické soubory jako obrázky, favicony, fonty, skripty a videa.
- **scripts/merge-builds.js** – Skript pro slučování buildů, vytvořený pro sestavení minifikovaných a neminifikovaných CSS a JS souborů pro optimalizaci minifikace kódu.
- **index.html** – Hlavní HTML soubor aplikace, do kterého se React aplikace renderuje.
- **Konfigurační soubory** – *vite.config.js*, *eslint.config.js*, *package.json*, *.env*, *.gitignore*, *netlify.toml* – slouží pro konfiguraci vývojového prostředí a nasazení aplikace.
- **Další soubory** - *README.md* k dokumentaci projektu, *.nvmrc* k určení preferované verze Node.js

7.2 Implementace projektu

Kapitola se zaměřuje na funkcionality aplikace, testovací moduly a ukázky zajímavých kódů aplikace.

7.2.1 Funkcionalita aplikace

Aplikace Web Optimizer poskytuje:

- **Interaktivní porovnání verzí** – možnost přepínat mezi neoptimalizovanými a optimalizovanými testovacími scénáři. Výkon je měřen dvěma způsoby: v reálném čase přímo v prohlížeči pomocí knihovny web-vitals (LCP, CLS, INP a další metriky) a komplexní analýzou přes PageSpeed Insights API. Stránky lze také testovat vložením jejich URL do externích měřících nástrojů.
- **Teoretické informace o optimalizačních technikách** – aplikace obsahuje podstránky s vysvětlením metod jako minifikace kódu, lazy loading, optimalizace obrázků, fontů, načítání skriptů, CDN, API optimalizace a využití cache.
- **Přehled Core Web Vitals** – detailní vysvětlení metrik Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS), Interaction to Next Paint (INP) a okrajové vysvětlení ostatních metrik Web Vitals.
- **Stránka s výsledky testování** – shrnutí dosažených výsledků testů, jejich interpretace a vysvětlení, jak jednotlivé faktory ovlivňují výkon webu.
- **Stránka FAQ** – odpovědi na zajímavé otázky o optimalizaci výkonu webů.
- **Přímé odkazy na nástroje pro měření výkonu** – (PageSpeed Insights, GTmetrix, WebPageTest, Pingdom Tools, PageSpeed.cz, SpeedCurve).
- **Inspirativní informace o optimalizaci webu** – reálné případové studie firem jako Pinterest, Vodafone nebo BBC o dopadu optimalizace na byznys výsledky.

7.2.2 Struktura testovacích modulů

Testovací moduly jsou implementovány jako samostatné React komponenty v adresáři **src/pages/Testing** a na URL adrese *https://web-optimizer.netlify.app/testovani*. Každý modul reprezentuje konkrétní optimalizační techniku nebo metriku Core Web Vitals.

Implementace přepínání mezi verzemi stránky

Systém přepínání mezi optimalizovanou a neoptimalizovanou verzí je realizován pomocí URL parametrů. Například v komponentě *CLSTesting.jsx* je využíván React Router hook *useSearchParams* pro získání a nastavení parametrů v URL. Při prvním načtení stránky *useEffect* zajišťuje nastavení výchozí hodnoty na *false*, pokud není v URL žádná hodnota definována. Funkcionalita je uživatelům zpřístupněna prostřednictvím přepínacího tlačítka, které spouští funkci *handleClsToggle*. Výsledná URL adresa pak obsahuje parametr *optimized* s hodnotou *true* nebo *false* (např. *https://web-optimizer.netlify.app/testovani/cls-testing?optimized=false*).

```
const CLSTesting = () => {
  const [searchParams, setSearchParams] = useSearchParams();
  const [clsValue, setClsValue] = useState("Čekání na měření...");
  const [activeTest, setActiveTest] = useState("gallery");

  // Při prvním načtení stránky nastavíme parametr "optimized" na false
  useEffect(() => {
    if (!searchParams.has("optimized")) {
      setSearchParams({ optimized: "false" }, { replace: true });
    }
  }, [searchParams, setSearchParams]);

  // Přepínání mezi optimalizovanou a neoptimalizovanou verzí
  const handleClsToggle = () => {
    setSearchParams({ optimized: !isOptimized ? "true" : "false" });
  };

  const isOptimized = searchParams.get("optimized") === "true";
```

Obr. 7: Implementace přepínání mezi verzemi stránky
Zdroj: vlastní zpracování (2025)

Monitoring Core Web Vitals metrik

Aplikace využívá knihovnu web-vitals pro měření výkonnostních metrik v reálném čase přímo v prohlížeči. Implementace zahrnuje sledování LCP, CLS, INP a dalších klíčových metrik, což umožňuje uživatelům okamžitě pozorovat dopad jednotlivých optimalizací.

```
// Měření Core Web Vitals
useEffect(() => {
  let isMounted = true;

  const handleMetric =
    (name) =>
    ({ value }) => {
      if (isMounted) {
        setMetrics((prev) => ({
          ...prev,
          [name]: name === "cls" ? value.toFixed(3) : value.toFixed(0),
        }));
      }
    };

  try {
    const unsubLCP = onLCP(handleMetric("lcp"));
    const unsubCLS = onCLS(handleMetric("cls"));
    const unsubINP = onINP(handleMetric("inp"));
    const unsubFCP = onFCP(handleMetric("fcp"));
    const unsubTTFB = onTTFB(handleMetric("ttfb"));

    return () => {
      isMounted = false;
      if (typeof unsubLCP === "function") unsubLCP();
      if (typeof unsubCLS === "function") unsubCLS();
      if (typeof unsubINP === "function") unsubINP();
      if (typeof unsubFCP === "function") unsubFCP();
      if (typeof unsubTTFB === "function") unsubTTFB();
    };
  } catch (error) {
    console.error("Chyba při měření metrik:", error);
  }
}, [isOptimized]);
```

Obr. 8: Monitoring Core Web Vitals metrik

Zdroj: vlastní zpracování (2025)

Adaptivní optimalizace obrazových souborů

V rámci optimalizační komponenty jsou implementovány tři různé techniky pro zlepšení načítání obrázků. Řešení dynamicky aplikuje kombinaci moderních formátů (JPG, WebP, AVIF), lazy loading a specifikaci rozměrů podle aktuálního uživatelského nastavení. Každý obrázek v galerii se vykresluje s parametry odpovídajícími stavu přepínačů. Specifikace atributů *width* a *height* eliminuje nežádoucí posuny layoutu (CLS), zatímco lazy loading zkracuje dobu načítání. Formáty WebP a AVIF pak přinášejí významnou redukci datové velikosti při zachování vizuální kvality.

```jsx
<div className="gallery section-page">
  <div className="gallery__container">
    {Array.from({ length: 15 }, (_, index) => (
      <img
        key={index}
        src={`/assets/images/image${index + 1}.${format}`}
        loading={isLazy ? "lazy" : "eager"}
        width={showAttributes ? "300" : undefined}
        height={showAttributes ? "200" : undefined}
        alt={`Obrázek ${index + 1}`}
        className="gallery__item"
      />
    ))}
  </div>
</div>
```

Obr. 9: Implementace obrázků s nastavením parametrů

Zdroj: vlastní zpracování (2025)

Optimalizované načítání YouTube videí

Pro zlepšení výkonu stránek s vloženými YouTube videi je vytvořena komponenta s odloženým načítáním. Místo okamžitého načtení iframe elementu se nejprve zobrazuje pouze náhledový obrázek videa. Teprve po interakci uživatele (kliknutí) dochází k načtení samotného iframe elementu s videem. Zmíněná technika významně snižuje zátěž při prvotním vykreslení stránky a pozitivně ovlivňuje metriky **LCP** a **INP**.

```jsx
const LazyYoutubeEmbed = ({ videoId, isOptimized }) => {
  const [isLoaded, setIsLoaded] = useState(false);

  return (
    <div
      className={`youtube-container ${
        isOptimized ? "" : "youtube-container--unoptimized"
      }`}
      onClick={() => setIsLoaded(true)}>
      {isLoaded || !isOptimized ? (
        <iframe
          width="100%"
          height="100%"
          src={`https://www.youtube.com/embed/${videoId}?autoplay=0&rel=0&modestbranding=1`}
          title="YouTube video"
          frameBorder="0"
          allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture"
          allowFullScreen
          loading={isOptimized ? "lazy" : "eager"}></iframe>
      ) : (
        <img
          src={`https://img.youtube.com/vi/${videoId}/hqdefault.jpg`}
          alt="YouTube video preview"
          className="youtube-preview"
        />
      )}
    </div>
  );
};

export default LazyYoutubeEmbed;
```

Obr. 10: Definice komponenty LayYoutubeEmbeded.jsx

Zdroj: vlastní zpracování (2025)

Integrace komponenty pro odložené načítání

Implementovaná komponenta *LazyYoutubeEmbed* je následně využívána v hlavní testovací stránce, kde dochází k iteraci přes seznam videí a jejich dynamickému vkládání na stránku. V optimalizované verzi dochází k načtení videa až po interakci uživatele, zatímco neoptimalizovaná varianta načítá iframe elementy bezprostředně při vykreslení stránky.

```
<section className="section-page">
  <h2 className="section-subtitle">
    Optimalizované načítání YouTube videí
  </h2>
  <p className="section-text">
    {isOptimized
      ? "YouTube video se nejprve zobrazí jako náhledový obrázek a iframe se nač
      : "Iframe s videem se načítá ihned, což může zpomalit načítání stránky."}
  </p>
  <div className="youtube__grid">
    {videoIds.map((videoId, index) => (
      <LazyYoutubeEmbed
        key={index}
        videoId={videoId}
        isOptimized={isOptimized}
      />
    ))}
  </div>
</section>
```

Obr. 11: Použití komponenty LazyYoutubeEmbed v hlavní testovací stránce

Zdroj: vlastní zpracování (2025)

Optimalizace interaktivity asynchronním zpracováním

Pro zlepšení metriky INP je implementována optimalizace interaktivity přes asynchronní zpracování náročných operací. Výpočty jsou v optimalizované verzi přesunuty mimo hlavní vlákno pomocí *setTimeout*, čímž se zachovává responzivita rozhraní.

```
// Simulace těžkého výpočtu (blokující výpočet vs. asynchronní zpracování)
const handleComputationClick = () => {
  setStatus("Výpočet probíhá...");

  if (isOptimized) {
    setTimeout(() => {
      const result = heavyComputation(40);
      setStatus(`Výpočet dokončen: ${result}`);
    }, 0); // Asynchronní zpracování
  } else {
    const result = heavyComputation(40);
    setStatus(`Výpočet dokončen: ${result}`); // Blokující výpočet
  }
};
```

Obr. 12: Optimalizace interaktivity asynchronním zpracováním

Zdroj: vlastní zpracování (2025)

7.3 Implementace PSI API

Pro automatické měření výkonu konkrétních stránek je využito Google PageSpeed Insights API a již výše zmíněná web-vitals knihovna.

7.3.1 Funkce pro volání PageSpeed Insights API

Funkce *fetchPageSpeedResults* zajišťuje komunikaci s Google PageSpeed Insights API a získání klíčových metrik výkonu webové stránky. Proces začíná sestavením URL požadavku obsahujícího testovanou adresu a API klíč. Následně je odeslán požadavek pomocí metody *fetch()*. Při úspěšné odpovědi dochází ke konverzi do formátu JSON a extrakci relevantních dat z objektu *lighthouseResult.audits*. Objekt poskytuje důležité metriky výkonu jako Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS) a Interaction to Next Paint (INP). V případě neúspěchu je vygenerována a zobrazena odpovídající chybová zpráva.

```
const fetchPageSpeedResults = async (url) => {
  // Vytvoření URL pro volání API
  const apiUrl = `https://www.googleapis.com/pagespeedonline/v5/runPagespeed?url=${encodeURIComponent(
    url
  )}&key=${import.meta.env.VITE_PAGESPEED_API_KEY}`;

  try {
    // Odeslání požadavku na API
    const response = await fetch(apiUrl);
    if (!response.ok) {
      throw new Error(`Chyba API: ${response.status} ${response.statusText}`);
    }

    // Zpracování odpovědi do formátu JSON
    const data = await response.json();
    return data.lighthouseResult.audits; // Vrácení klíčových metrik z odpovědi API
  } catch (error) {
    throw new Error(error.message || "Nepodařilo se načíst data.");
  }
};
```

Obr. 13: Funkce pro volání PageSpeed Insights API

Zdroj: vlastní zpracování (2025)

7.3.2 Spuštění testu a zpracování výsledků

Implementovaná funkce *testPage* představuje interakční vrstvu mezi uživatelem a API. Po aktivaci uživatelského rozhraní dochází k nastavení stavu načítání a odeslání požadavku na API prostřednictvím dříve popsané funkce *fetchPageSpeedResults*. Po zpracování odpovědi jsou relevantní metriky (LCP, CLS, INP) uloženy do stavu *result* pro následné zobrazení v uživatelském rozhraní. Ošetřeny jsou i případné chybové stavy, které jsou komunikovány uživateli prostřednictvím odpovídající notifikace.

```
const testPage = async (url) => {
  setLoading(true);
  setError(null);
  setResult(null);

  try {
    const audits = await fetchPageSpeedResults(url);
    setResult(audits);
  } catch (error) {
    setError(error.message);
  } finally {
    setLoading(false);
  }
};
```

Obr. 14: Funkce pro spuštění testu

Zdroj: vlastní zpracování (2025)

7.3.3 Zobrazení výsledků testu

Uživatelské rozhraní pro testování obsahuje prvky pro iniciaci testu a prostor pro prezentaci výsledků ve formě přehledného seznamu. Výstup zahrnuje hodnoty klíčových metrik Core Web Vitals, které poskytují jasný obraz o výkonnostním profilu testované stránky. Údaje jsou zásadní pro identifikaci problémových oblastí a formulaci konkrétních optimalizačních strategií. V případě selhání procesu testování je uživatel informován prostřednictvím odpovídajícího chybového hlášení.

```
    <button
      className="button -bottom"
      onClick={() => testPage(window.location.href)}
      disabled={loading}>
      {loading ? <span>Testuji stránku...</span> : "Otestovat stránku"}
    </button>

    {error && <p className="test-page-speed__error">{error}</p>}

    {result && (
      <div className="test-page-speed__results">
        <h3>Výsledky testu (PageSpeed Insights)</h3>
        <ul>
          <li>
            <strong>LCP (Largest Contentful Paint):</strong>{" "}
            {result["largest-contentful-paint"]?.displayValue || "N/A"} –
            Rychlost vykreslení největšího prvku
          </li>
          <li>
            <strong>CLS (Cumulative Layout Shift):</strong>{" "}
            {result["cumulative-layout-shift"]?.displayValue || "N/A"} – Míra
            vizuální stability
          </li>
          <li>
            <strong>INP (Interaction to Next Paint):</strong>{" "}
            {result["interactive"]?.displayValue || "N/A"} – Rychlost
            interakce
          </li>
        </ul>
      </div>
    )}
  </div>
);
```

Obr. 15: Tlačítko a zobrazení výsledků testu

Zdroj: vlastní zpracování (2025)

8 Analýza optimalizačních technik

Cílem kapitoly je ověřit efektivitu jednotlivých optimalizačních technik a jejich vliv na metriky výkonu. Pro měření jsou využity různé nástroje a analýza probíhá v jednotném testovacím prostředí.

8.1 Metodologie testování

Pro objektivní zhodnocení vlivu optimalizačních technik je stanovena jasná metodologie testování. Testování probíhá na webové aplikaci vyvinuté v rámci práce, která umožňuje vyhodnocovat různé metriky výkonu v odlišných optimalizačních scénářích.

8.1.1 Testovací prostředí

Testy jsou prováděny v jednotném prostředí pro dosažení konzistentních výsledků:

- **Zařízení**: Notebook Dell s procesorem Intel Core i5, 16 GB RAM, SSD disk.
- **Operační systém**: Windows 10.
- **Prohlížeč**: Brave (založený na Chromium).
- **Síťové připojení**: Stabilní internetové připojení s rychlostí 60 Mb/s.

8.1.2 Použité nástroje

Pro sledování metrik výkonu jsou použity následující nástroje:

- **Web-vitals knihovna** – měření Core Web Vitals metrik přímo v prohlížeči v reálném čase.
- **PageSpeed Insights API** – automatizované měření metrik Core Web Vitals v aplikaci.
- **PageSpeed Insights** – komplexní analytický nástroj, se kterým se doporučuje začít.
- **GTmetrix** – podrobnější analýza výkonu a doporučení pro optimalizaci.
- **WebPageTest** – testy vykreslování v různých podmínkách.
- **Chrome DevTools** – detailní sledování načítání a profilování výkonu.

8.1.3 Postup testování

Každá testovaná stránka je analyzována dvakrát:

1. **Neoptimalizovaná verze** – základní test bez aplikovaných optimalizačních technik.
2. **Optimalizovaná verze** – stejný test po zavedení optimalizačních metod.

Každé měření proběhne několikrát, přičemž jsou zaznamenány průměrné hodnoty.

8.2 Výsledky testování optimalizací

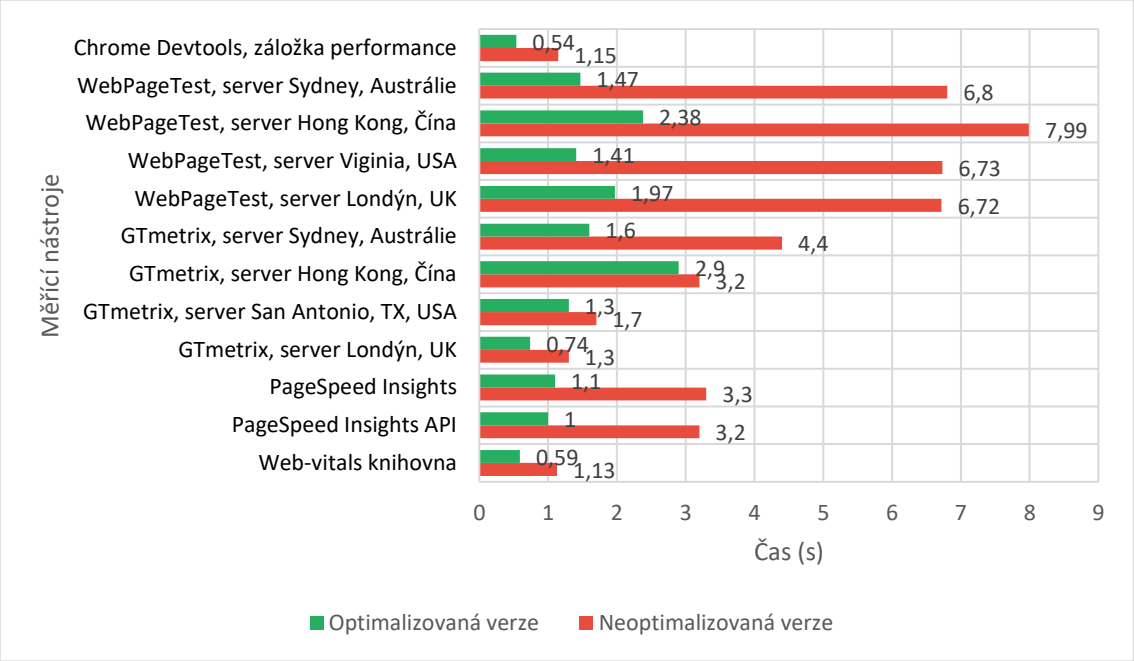
V sekci jsou prezentovány výsledky testování vybraných optimalizačních technik s důrazem na metriky Core Web Vitals. Veškeré testování probíhá na aplikaci dostupné na adrese

https://web-optimizer.netlify.app/testovani. Každá testovací stránka umožňuje přepínání mezi optimalizovanou a neoptimalizovanou verzí prostřednictvím *query* parametrů na konci adresy (např. *?optimized=true*).

8.2.1 Stránka na LCP test

Měření hodnot LCP probíhá na specializované testovací stránce aplikace. Klíčovým LCP prvkem je hlavní hero obrázek, jehož zpracování významně ovlivňuje celkovou rychlost načítání.

Graf 1: LCP testy na desktopu v různých nástrojích



Zdroj: vlastní zpracování (2025)

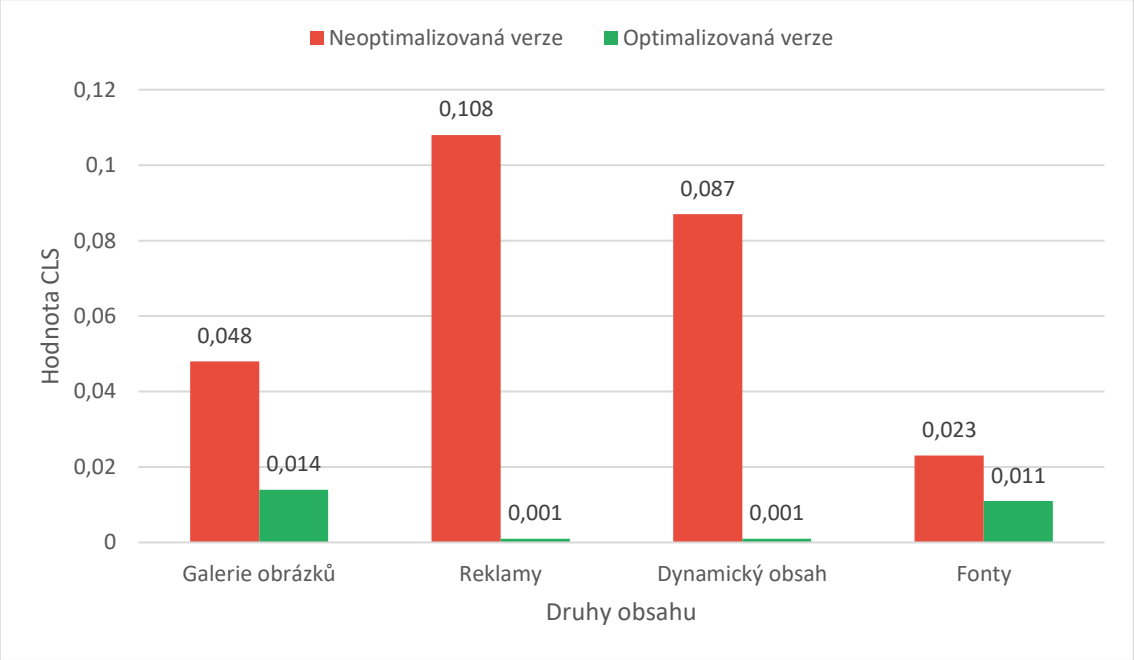
Z výsledků je patrný rozdíl v LCP hodnotách mezi verzemi stránky. Nejdramatičtější zlepšení je vidět u WebPageTest, kde optimalizace snížila LCP až o 79 % (z 6,73 s na 1,41 s na serveru ve Virginii). Důvodem výrazného rozdílu je pravděpodobně kombinace několika faktorů. WebPageTest provádí testy v kontrolovaném prostředí s omezenou šířkou pásma a na standardizovaných zařízeních, což umožňuje lépe demonstrovat dopady optimalizací. Navíc WebPageTest simuluje úplně první načtení stránky bez jakéhokoliv cachování, které v reálném světě často maskuje problémy s výkonem.

Geografická lokace testovacích serverů hraje rovněž významnou roli ve výsledcích. Přestože je aplikace hostována na Netlify, které má CDN síť po celém světě, jsou viditelné rozdíly mezi regiony. Zajímavé je, že největší zlepšení vykazují testy z USA a Austrálie, zatímco nejmenší rozdíl je z Hong Kongu (pouze 9,4 % u GTmetrix). Příčinou může být síťová konektivita nebo zatížení serverů v daném regionu. Velikost WebP obrázku v optimalizované verzi (vs. JPEG v neoptimalizované) výrazně snižuje dobu přenosu souboru. Efekt redukce velikosti je nejvíce patrný při testování na velké vzdálenosti, kde hraje značnou roli latence.

8.2.2 Stránka na CLS test

Analýza Cumulative Layout Shift zahrnuje čtyři různé scénáře způsobující posuny obsahu: galerii obrázků, reklamy, dynamický obsah a webové fonty. Měření je realizováno pomocí knihovny web-vitals v prostředí standardizovaného testovacího rozhraní.

Graf 2: CLS testy pro různé typy obsahu na desktopu



Zdroj: vlastní zpracování (2025)

Výsledky ukazují podstatné zlepšení CLS u všech testovaných prvků. Největší posun je zaznamenán u reklam (99,1 %) a dynamického obsahu (98,9 %), kde optimalizace téměř eliminovala CLS. Důvodem je především výška reklamního banneru (1000px), která bez optimalizace způsobuje masivní posun okolního obsahu. U galerie obrázků zajistilo explicitní definování atributů *width* a *height* zlepšení o 70,8 %. Fonty vykazují nejmenší, ale stále zřetelné, zlepšení (52,2 %), což je výsledkem využití *font-display: swap* a konzistentních rozměrů textu.

8.2.3 Stránka na INP test

Hodnocení metriky Interaction to Next Paint je zaměřeno na dva scénáře uživatelské interakce: rekurzivní výpočet Fibonacciho posloupnosti a zpracování formuláře s náročnou validací. Oba případy představují typické situace, kdy může docházet k blokování hlavního vlákna prohlížeče.

Tab. 2: INP testy – Fibonacciho posloupnost a odeslání formuláře

Test	INP neoptimalizované (ms)	INP optimalizované (ms)	Zlepšení (%)
Fibonacci výpočet	3049,80	0,10	99,99
Odeslání formuláře	1013,00	0,10	99,99

Zdroj: vlastní zpracování (2025)

Na rozdíl od ostatních testů, INP nebylo měřeno pomocí web-vitals knihovny, ale přímo pomocí *performance.now() API*. Důvodem je, že INP vyžaduje skutečnou interakci uživatele (kliknutí, psaní) a nelze ji snadno automatizovat.

Výsledky ukazují značný rozdíl mezi oběma implementacemi. U neoptimalizované verze trvá výpočet Fibonacciho čísla více než 3 sekundy a během této doby je uživatelské rozhraní zcela zablokované. Podobně u formuláře trvá validace přes 1 sekundu. Obě hodnoty hodně překračují hranici 500 ms pro špatnou INP, což by v reálném prostředí vedlo k velmi špatné uživatelské zkušenosti.

Optimalizovaná verze používá asynchronní zpracování pomocí *setTimeout* s nulovým zpožděním, což přesouvá náročný výpočet mimo hlavní vlákno. Díky tomu je doba od interakce k prvnímu překreslení pouhých 0,10 ms. Uživatel tak dostává okamžitou vizuální odezvu a může pokračovat v práci s aplikací, zatímco výpočet probíhá na pozadí.

8.2.4 Testování vlivu CDN

Srovnání rychlosti načítání JavaScriptových knihoven využívá měření z různých CDN poskytovatelů (cdnjs, jsDelivr) a lokálního hostingu. Analýza probíhá v anonymním okně prohlížeče bez zakázání cache, což umožňuje zachytit rozdíly mezi prvním načtením a cachovanou verzí.

Tab. 3: Rychlost načítání knihoven z různých zdrojů (první načtení vs. cachované)

Knihovna	Zdroj	Fetch 1. načtení (ms)	Onload 1. načtení (ms)	Fetch z cache (ms)	Onload z cache (ms)	Velikost (kB)
Lodash.js	cdnjs	81,10	77,50	30,30	9,30	71,30
Lodash.js	jsDelivr	86,90	61,50	49,60	8,70	71,30
Lodash.js	Lokální	256,00	62,90	26,30	35,50	71,30
Chart.js	jsDelivr	31,10	46,10	29,80	20,30	201,06
Chart.js	cdnjs	30,60	91,80	40,40	20,00	183,95
Chart.js	Lokální	375,70	43,60	29,30	36,90	201,06
TensorFlow.js	jsDelivr	41,00	208,90	24,20	176,80	1427,01
TensorFlow.js	cdnjs	758,70	229,20	28,20	136,80	1427,01
TensorFlow.js	Lokální	229,70	195,60	53,90	168,80	1427,01
Babylon.js	jsDelivr	694,50	170,00	50,80	170,50	4718,33
Babylon.js	cdnjs	88,30	409,90	30,10	46,50	4667,30
Babylon.js	Lokální	274,10	441,50	24,10	220,60	4718,33

Zdroj: vlastní zpracování (2025)

Výsledky ukazují stěžejní rozdíly mezi prvním načtením a cachovaným načtením, což demonstruje důležitost správného nastavení cache headers pro optimalizaci výkonu webových aplikací. Při prvním načtení jsou CDN obecně rychlejší než lokální hosting pro menší

knihovny (Lodash.js, Chart.js), zatímco pro velké knihovny (TensorFlow.js, Babylon.js) může být lokální hosting efektivnější.

Zajímavým zjištěním je rozdíl v době načítání Babylon.js (694,50 ms z jsDelivr vs. 274,10 ms z lokálního hostingu). To může být způsobeno momentálním vytížením CDN nebo geografickou vzdáleností k CDN serverům. Opakované měření potvrdilo kolísání výkonu CDN, kde časy prvního načtení vykazovaly značnou variabilitu. Při cachovaném načtení jsou však výkonnostní rozdíly mezi CDN a lokálním hostingem minimální. U větších knihoven jako TensorFlow.js a Babylon.js zůstává doba zpracování (onload) vysoká i při cachovaném načtení, což naznačuje, že samotná velikost a složitost JavaScriptu představuje výraznější omezení výkonu než přenos souboru.

Z provedených měření vyplývá, že rozdíly mezi CDN a lokálním hostingem závisí zejména na velikosti knihovny a charakteru cílové skupiny. Zatímco menší knihovny se mohou z CDN načítat o něco rychleji, lokální hosting přináší výhody z hlediska bezpečnosti, nezávislosti na třetích stranách a plné kontroly nad obsahem. U větších knihoven se navíc může projevit vyšší efektivita lokálního hostingu. Významnou roli hraje také cachování, které snižuje rozdíly při opakovaných návštěvách – což zdůrazňuje důležitost správného nastavení HTTP cache headers. Při volbě způsobu distribuce by mělo být zohledněno geografické rozložení uživatelů, poměr nových a vracejících se návštěvníků a požadavky na spolehlivost.

8.2.5 Testování výkonu API

Komparativní analýza veřejných API zahrnuje měření rychlosti odezvy, velikosti přenesených dat a efektivity komprese. Každý endpoint je testován opakovaně pro vyhodnocení vlivu cachování na celkový výkon.

Tab. 4: Výsledky testování výkonu vybraných API

API	Velikost (kB)	První TTFB (ms)	Druhý TTFB (ms)	Zlepšení (%)	Komprese (%)	Cache status
Crypto Prices	0,05	257,50	17,50	92,8	žádná	Povoleno (30 s)
Dog Breeds	2,34	200,20	14,50	92,3	56,8	Povoleno (1800 s)
GitHub Users	30,16	234,10	12,70	94,0	žádná	Povoleno (60 s)
Countries (Detailed)	161,57	456,70	16,10	96,1	79,2	Povoleno (31556926 s)
Pokemon (1000 záznamů)	65,33	295,40	14,30	94,6	87,0	Povoleno (86400 s)
Random Users (1000)	1055,36	589,30	401,70	31,8	žádná	Zakázáno
Cat Fact	0,17	236,40	154,10	34,7	žádná	Zakázáno
Programming Books	20,40	2671,40	1214,90	54,5	žádná	Nenastaveno

Zdroj: vlastní zpracování (2025)

Z výsledků testování lze vyčíst rozdíl v rychlosti odezvy mezi prvním a druhým voláním API. U endpointů s povoleným cachováním je zlepšení výrazně vyšší (průměrně 93,1 %) oproti API bez cachování (průměrně 44,7 %). Nejlepších výsledků dosáhl endpoint Countries s kompresí 79,2 % a zlepšením času odezvy o 96,1 %.

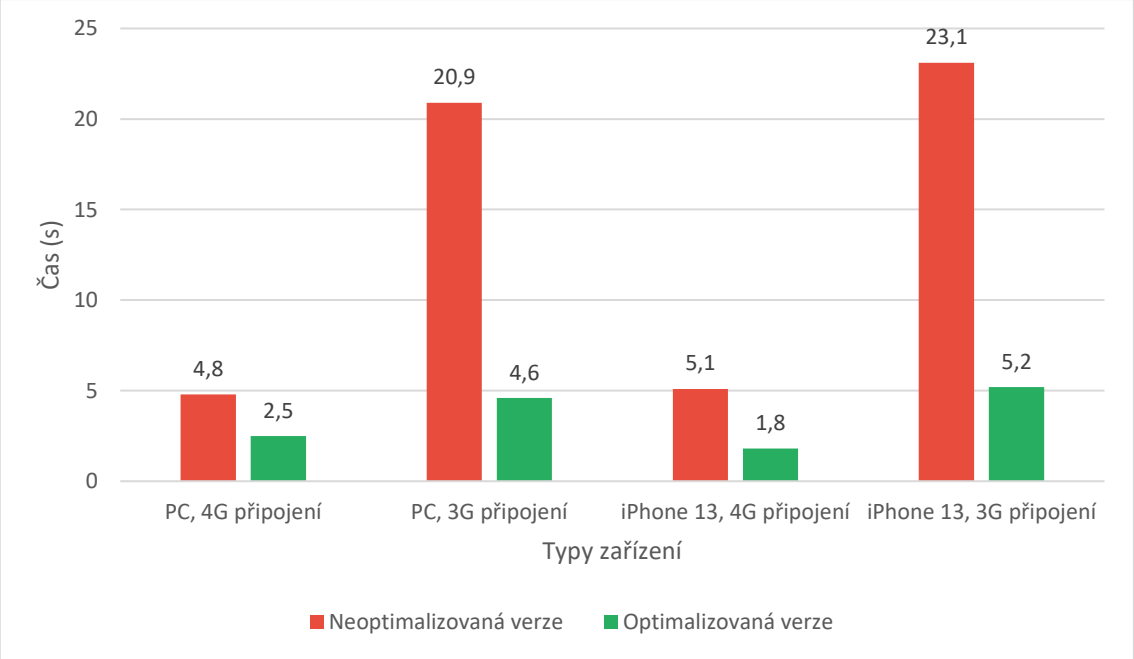
Zajímavé je také sledovat vztah mezi velikostí dat a mírou komprese. Největší kompresi vykazuje Pokemon API (87 %), což výrazně zrychluje přenos dat u tohoto relativně velkého datasetu.

U některých API je zřetelné zlepšení i bez explicitně povoleného cachování na straně serveru. Je to způsobeno tím, že prohlížeč sám ukládá odpovědi do své diskové cache, což velmi urychluje opakované načítání stejných dat i bez specifických cache hlaviček nastavených na serveru.

8.2.6 Testování vlivu rychlosti internetu

Hodnocení dopadu různých síťových podmínek na metriku LCP využívá nástroj GTmetrix, který umožňuje simulovat různé rychlosti připojení. Cílem je kvantifikovat význam optimalizací při odlišných síťových omezeních.

Graf 3: Vliv rychlosti připojení na LCP



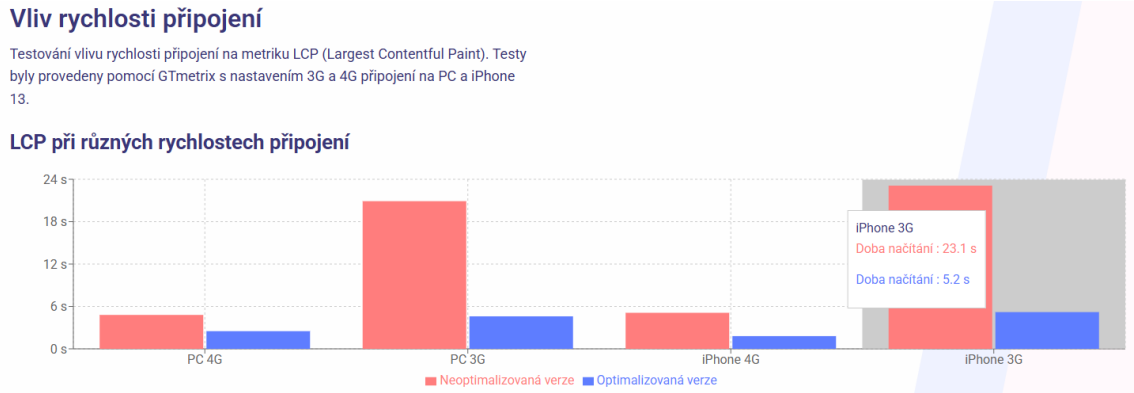
Zdroj: vlastní zpracování (2025)

Výsledky demonstrují, že rychlost připojení má výrazný dopad na LCP, zejména u neoptimalizované verze. Při přechodu z 4G na 3G se LCP u neoptimalizované verze zhoršilo více než čtyřnásobně na PC (z 4,8 s na 20,9 s) a podobně na iPhonu 13 (z 5,1 s na 23,1 s).

Optimalizovaná verze vykazuje mnohem menší citlivost na změnu rychlosti připojení. I na pomalém 3G připojení dosahuje přijatelných hodnot kolem 5 sekund. Relativní zlepšení díky optimalizacím je výraznější na pomalejších připojeních (77–78 % na 3G oproti 48–65 % na 4G), což potvrzuje, že webové optimalizace jsou nejdůležitější právě pro uživatele s pomalejším připojením.

8.3 Závěry z testování optimalizačních technik

Z provedené analýzy vyplývá několik praktických poznatků s dopadem na výkon webových aplikací. Asynchronní načítání JavaScriptu přineslo téměř 100% zlepšení metriky INP a výrazně zvýšilo interaktivitu stránek. Optimalizace obrázků pomocí formátu WebP zkrátila dobu načítání hlavního obsahu (LCP) až o 79 %, přičemž efekt závisel na velikosti původního neoptimalizovaného souboru. Výrazné zlepšení metriky CLS (až o 98 %) bylo dosaženo rezervací prostoru pro dynamické prvky, zejména u uměle vytvořených reklam, kde jinak docházelo k výrazným posunům obsahu. Cachování API odpovědí zrychlilo opakované načítání stránek až o 96 %. Testování zároveň ukázalo, že optimalizační techniky mají největší relativní přínos při pomalém připojení (např. 3G). Z hlediska distribuce knihoven se jako vhodnější řešení pro větší soubory ukázal lokální hosting, který nabízí větší kontrolu, nezávislost a bezpečnost oproti externím CDN.



Obr. 16: Vliv rychlosti připojení na metriku LCP
Zdroj: vlastní zpracování (2025)

Výsledky měření z testovací aplikace na stránce */vysledky*, provedené pomocí GTmetrix s nastavením 3G a 4G připojení na PC a iPhonu 13. Graf ukazuje, že optimalizovaná verze výrazně zkracuje dobu načítání hlavního obsahu, zejména při pomalém připojení (např. z 23,1 s na 5,2 s u iPhonu na 3G).

9 Uživatelské testování různých podmínek

Kapitola se zaměřuje na testování výkonnosti webových aplikací v reálných podmínkách na různých zařízeních. Na rozdíl od předchozí části, která analyzovala jednotlivé optimalizační techniky, zde je cílem vyhodnotit, jak hardware, rychlost připojení a další faktory ovlivňují uživatelskou zkušenost a efektivitu implementovaných optimalizací.

9.1 Metodika testování

Analýza je realizována na komplexní testovací stránce obsahující široké spektrum komponent typických pro moderní webové aplikace – od velkých hero obrázků po interaktivní prvky a externí obsah. Každé měření probíhá na optimalizované i neoptimalizované verzi. Výkonnostní metriky jsou získávány pomocí knihovny web-vitals implementované přímo v aplikaci a prostřednictvím Lighthouse testů pro simulaci standardizovaných podmínek.

9.2 Srovnání výkonu na různých reálných zařízeních

Pro komplexní analýzu vlivu hardwaru na výkon webových aplikací bylo vybráno několik zařízení pokrývajících široké spektrum výkonnostních kategorií. Testování zahrnuje počítače různých výkonnostních tříd a mobilní zařízení odlišných výrobců a cenových kategorií. U každého zařízení jsou měřeny hodnoty Core Web Vitals na obou verzích testovací stránky.

Testovaná zařízení:

- **Počítače** (seřazeno od nejvýkonnějšího):
    - MacBook Air M3 (16GB RAM, SSD) – prémiový výkon s ARM architekturou
    - Dell (16GB RAM, SSD, Intel Core i5) – středně výkonná konfigurace
    - Starý PC (4GB RAM, HDD, starší Intel procesor) – reprezentant základního hardware
- **Mobilní zařízení** (seřazeno od nejvýkonnějšího):
    - iPhone 13 – zástupce prémiového segmentu s procesorem A15 Bionic
    - Xiaomi Redmi Note 10 – reprezentant střední třídy Android zařízení
    - Huawei P20 lite – starší Android zařízení

9.2.1 Vliv hardwaru počítače

Výkonnost webových aplikací byla posuzována na základě měření na třech vybraných počítačích s rozdílnou úrovní výkonu. Cílem bylo zjistit, jak se liší hodnoty Core Web Vitals v závislosti na použité hardwarové konfiguraci. Data jsou získána kombinací měření pomocí integrované web-vitals knihovny a nástroje Lighthouse pro zajištění komplexního pohledu.

Tab. 5: Hodnoty naměřené pomocí web-vitals knihovny

Zařízení	Verze	LCP (ms)	CLS	INP (ms)
Dell (16GB RAM, SSD)	Neoptimalizovaná	3 360	0,332	2 376
Dell (16GB RAM, SSD)	Optimalizovaná	448	0,014	64
Zlepšení		**86, 7 %**	**95,8 %**	**97,3 %**
MacBook (M3, 16GB RAM)	Neoptimalizovaná	2 692	0,33	688
MacBook (M3, 16GB RAM)	Optimalizovaná	420	0,001	48
Zlepšení		**84,4 %**	**99,7 %**	**93 %**
Starý PC (4GB RAM, HDD)	Neoptimalizovaná	3 790	0,42	5 776
Starý PC (4GB RAM, HDD)	Optimalizovaná	1 264	0,03	464
Zlepšení		**66,7 %**	**92,9 %**	**92 %**

Zdroj: vlastní zpracování (2025)

Tab. 6: Hodnoty naměřené pomocí Lighthouse

Zařízení	Verze	LCP (ms)	CLS	INP (ms)
Dell (16GB RAM, SSD)	Neoptimalizovaná	10 100	0,273	1 060
Dell (16GB RAM, SSD)	Optimalizovaná	448	0,014	64
Zlepšení		**95,6 %**	**94,9 %**	**94,0 %**
MacBook (M3, 16GB RAM)	Neoptimalizovaná	8 500	0,367	710
MacBook (M3, 16GB RAM)	Optimalizovaná	900	0,003	32
Zlepšení		**89,4 %**	**99,2 %**	**95,5 %**
Starý PC (4GB RAM, HDD)	Neoptimalizovaná	15 500	0,45	6 000
Starý PC (4GB RAM, HDD)	Optimalizovaná	2 300	0,04	500
Zlepšení		**85,2 %**	**91,1 %**	**91,7 %**

Zdroj: vlastní zpracování (2025)

Z výsledků jasně vyplývá, že hardwarová konfigurace počítače má velký vliv na výkonnostní metriky, zejména na INP. Starý počítač s HDD dosahuje více než dvojnásobných hodnot INP (5 776 ms) proti středně výkonnému PC Dell (2 376 ms) při neoptimalizované verzi. Zajímavé je, že výkonný MacBook s čipem M3 dosahuje při neoptimalizované verzi hodnoty jen 688 ms.

LCP hodnoty vykazují menší závislost na hardwaru. U starého PC a středně výkonného PC Dell jsou relativně srovnatelné při měření web-vitals knihovnou. To naznačuje, že LCP je více ovlivněno rychlostí připojení než výkonem procesoru. Lighthouse testy však ukazují výraznější rozdíly, kde starý PC dosahuje až 15 500 ms LCP oproti 10 100 ms u středně výkonného PC Dell.

CLS hodnoty jsou mírně horší na starším hardware, ale rozdíly nejsou tak markantní jako u ostatních metrik. To potvrzuje, že vizuální stabilita závisí spíše na implementaci layoutu, nikoli na výkonu zařízení.

Nejdůležitějším zjištěním je, že optimalizace přinášejí konzistentní zlepšení napříč všemi zařízeními, s relativním zlepšením typicky 85–99 %. To znamená, že investice do optimalizací se vyplatí bez ohledu na cílovou skupinu uživatelů.

9.2.2 Porovnání mobilních zařízení s PC

Komparativní analýza mobilních platforem a osobních počítačů se zaměřuje na metriku INP. Testování probíhá na dvou typech interaktivních scénářů – rekurzivním výpočtu Fibonacci posloupnosti a zpracování formuláře s validací, které reprezentují typické výpočetně náročné operace v reálných aplikacích.

Tab. 7: Hodnoty INP na mobilních zařízeních

Zařízení	Test	Neoptimalizovaná verze (ms)	Optimalizovaná verze (ms)	Zlepšení (%)
iPhone 13	Fibonacci výpočet	1 432,00	0,01	99,99
iPhone 13	Formulář	672,00	0,01	99,99
Huawei P20 lite	Fibonacci výpočet	17 658,00	0,40	99,99
Huawei P20 lite	Formulář	7 556,00	0,40	99,99
Xiaomi Redmi Note 10	Fibonacci výpočet	7 426,60	0,10	99,99
Xiaomi Redmi Note 10	Formulář	2 744,40	0,10	99,99

Zdroj: vlastní zpracování (2025)

Tab. 8: Hodnoty INP na počítačích

Zařízení	Test	Neoptimalizovaná verze (ms)	Optimalizovaná verze (ms)	Zlepšení (%)
Dell (16GB RAM, SSD)	Fibonacci výpočet	2 523,50	0,20	99,99
Dell (16GB RAM, SSD)	Formulář	952,00	0,10	99,99
MacBook (M3, 16GB RAM)	Fibonacci výpočet	1 739,00	0,10	99,99
MacBook (M3, 16GB RAM)	Formulář	571,30	0,10	99,99
Starý PC (4GB RAM, HDD)	Fibonacci výpočet	6 742,40	0,20	99,99
Starý PC (4GB RAM, HDD)	Formulář	2 574,90	0,10	99,99

Zdroj: vlastní zpracování (2025)

Výsledky testování ukazují velké rozdíly mezi výkonností mobilních zařízení a PC při zpracování náročných JavaScriptových úloh v neoptimalizované verzi. Nejnápadnější je extrémní hodnota u telefonu Huawei P20 lite, kde Fibonacci výpočet trvá přes 17,6 sekund.

Na mobilních zařízeních je patrný významný rozdíl mezi iPhone 13 a průměrnými Android telefony. iPhone 13 dosahuje hodnot INP srovnatelných s výkonným MacBookem, zatímco Android zařízení střední třídy jsou výrazně pomalejší.

Zajímavé je srovnání starého PC (HP s 4GB RAM a HDD) s mobilními zařízeními. Přestože je uvedený počítač výrazně starší, stále dosahuje lepších hodnot než průměrné Android telefony. To ukazuje na významné výkonnostní omezení mobilních procesorů při zpracování komplexního JavaScriptu.

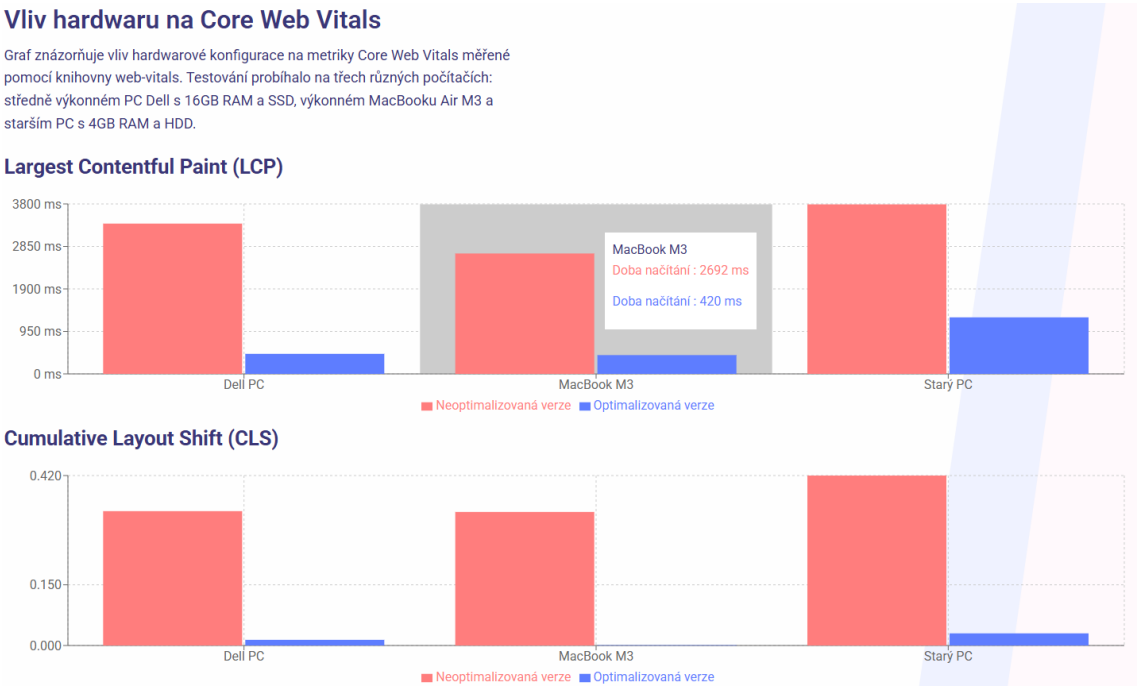
Nejdůležitějším zjištěním je téměř dokonalá optimalizace ve všech případech. Optimalizovaná verze s asynchronním zpracováním dosahuje na všech zařízeních hodnot pod 500 ms, což prakticky eliminuje jakékoli rozdíly mezi zařízeními.

Z výsledků vyplývá, že mobilní zařízení, zejména střední a nižší třídy, vyžadují mnohem důslednější optimalizace než desktopové počítače. Zatímco na výkonném PC může být neoptimalizovaný kód „pouze“ pomalý, na mobilním zařízení může způsobit úplnou nepoužitelnost aplikace.

9.3 Závěry z uživatelského testování

Výsledky testování na různých hardwarových platformách potvrzují zásadní význam webových optimalizací napříč celým spektrem zařízení. Největší rozdíly se projevily u metriky INP, kde neoptimalizovaná implementace vedla k až několikasekundovým prodlevám na méně výkonných zařízeních. Asynchronní zpracování JavaScriptu se ukázalo jako nejúčinnější technika s výrazným pozitivním dopadem.

Pro zajištění konzistentní uživatelské zkušenosti je nezbytné přizpůsobit optimalizační strategie cílovým zařízením – na mobilních platformách eliminovat blokující JavaScript a při pomalém připojení minimalizovat objem přenášených dat.



Obr. 17: Vliv hardwaru na metriky LCP a CLS

Zdroj: vlastní zpracování (2025)

Výstup z testovací aplikace na stránce */vysledky*, získaný pomocí knihovny web-vitals na třech počítačích s různou výkonností: Dell PC, MacBook Air M3 a starší PC s HDD. Grafy ukazují, že optimalizace zlepšuje výkon napříč zařízeními a může mít větší dopad než samotná hardwarová výbava.

Závěr

Cílem bakalářské práce bylo analyzovat dopad optimalizačních technik na výkon webových aplikací, ověřit účinnost a poskytnout praktická doporučení pro vývojáře. Cíl byl naplněn teoretickým rozborem problematiky i praktickou implementací testovací aplikace.

V teoretické části byly představeny způsoby měření výkonu webových aplikací zahrnující syntetická měření i analýzu dat reálných uživatelů. Důraz byl kladen na metriky Core Web Vitals (LCP, CLS a INP), které přímo ovlivňují uživatelskou zkušenost. Analýza nástrojů jako PageSpeed Insights, Lighthouse, WebPageTest a GTmetrix ukázala specifické výhody při diagnostice výkonnostních problémů. Práce popsala osvědčené optimalizační techniky včetně minifikace kódu, lazy loadingu, komprese obrazových souborů, optimalizace API volání a využití CDN.

Praktická část přinesla implementaci webové aplikace v Reactu pro objektivní porovnání optimalizovaných a neoptimalizovaných verzí komponent. Testování na různých zařízeních a při různých rychlostech připojení prokázalo, že optimalizace mají největší dopad právě u uživatelů se slabšími zařízeními a pomalejším připojením. Nejúčinnější optimalizační technikou se ukázalo asynchronní zpracování JavaScriptu, které výrazně zlepšilo metriku INP. Optimalizace obrázků pomocí moderních formátů zkrátila dobu načítání hlavního obsahu, zatímco rezervace prostoru pro dynamické prvky zlepšila hodnotu CLS.

Hlavním přínosem práce je zhodnocení dopadu jednotlivých optimalizačních technik v různých podmínkách a vytvoření praktické demonstrační aplikace, která slouží jako edukativní nástroj pro vývojáře. Výsledky testování potvrdily hypotézu, že relativní zlepšení díky optimalizacím je výraznější při pomalejším připojení a na méně výkonných zařízeních.

V budoucím výzkumu se nabízí podrobnější porovnání výkonnosti moderních frameworků, analýza vlivu server-side renderingu nebo automatizace optimalizací pomocí nástrojů založených na strojovém učení. Zajímavou oblastí je také zkoumání korelace mezi Core Web Vitals a obchodními metrikami jako jsou konverze či míra opuštění stránky.

Seznam použité literatury

ALEXANDREA, Jordana. *How to Run a Website Speed Test with GTmetrix*. Online. Hostinger. 2023. Dostupné z: https://www.hostinger.com/tutorials/gtmetrix-for-testing-websites-speed. [cit. 2024-12-09].

CLOUDFLARE. What is a content delivery network (CDN)? | How do CDNs work? Online. Cloudflare. 2024. Dostupné z: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/. [cit. 2024-12-22].

COPES, Flavio. *React for Beginners – A React.js Handbook for Front End Developers*. Online. FreeCodeCamp. 2020. Dostupné z: https://www.freecodecamp.org/news/react-beginner-handbook/. [cit. 2024-12-28].

COPES, Flavio. *The Express + Node.js Handbook – Learn the Express JavaScript Framework for Beginners*. Online. FreeCodeCamp. 2022. Dostupné z: https://www.freecodecamp.org/news/the-express-handbook/. [cit. 2024-12-28].

EVERTS, Tammy. 10 things I love about SpeedCurve (that I think you'll love, too). Online. SpeedCurve. 2023. Dostupné z: https://www.speedcurve.com/blog/10-things/. [cit. 2024-12-10].

GOOGLE FOR DEVELOPERS. *About PageSpeed Insights*. Online. PageSpeed Insights. 2024. Dostupné z: https://developers.google.com/speed/docs/insights/v5/about. [cit. 2024-11-30].

GRIFFIN, Jordan. *Improving Your Interaction to Next Paint (INP)*. Online. Request Metrics. 2024. Dostupné z: https://requestmetrics.com/web-performance/inp-interaction-to-next-paint/. [cit. 2024-11-18].

HANSA, Umar. *Synthetic Monitoring vs. Real User Monitoring*. Online. DebugBear. 2024. Dostupné z: https://www.debugbear.com/blog/synthetic-vs-rum. [cit. 2024-11-23].

HAWORTH, Hugh. *Comparing the New Generation of Build Tools*. Online. CSS-TRICKS. 2022. Dostupné z: https://css-tricks.com/comparing-the-new-generation-of-build-tools/. [cit. 2024-12-20].

HOLCOMBE, Jeremy. *A Complete Guide to Using WebPageTest (and Interpreting the Results)*. Online. Kinsta. 2024. Dostupné z: https://kinsta.com/blog/webpagetest/. [cit. 2024-12-10].

JACKSON, Brian. Pingdom Speed Test Tool: Ultimate Guide. Online. Kinsta. 2024. Dostupné z: https://kinsta.com/blog/pingdom-speed-test/. [cit. 2024-12-10].

JAHODA, Bohumil. *K čemu slouží CDN?* Online. Je čas. 2016. Dostupné z: https://jecas.cz/cdn. [cit. 2024-12-22].

KAREL, Arjen. *Improve Interaction to Next Paint (INP)*. Online. Core Web Vitals and Pagespeed consultancy. 2024. Dostupné z: https://www.corewebvitals.io/core-web-vitals/interaction-to-next-paint. [cit. 2024-11-18].

KEETON, B.J. *Speed Index: What it is & How to Optimize your Website for it*. Online. Elegant Themes. 2023. Dostupné z: https://www.elegantthemes.com/blog/wordpress/speed-index. [cit. 2024-11-28].

LAZARIS, Louis. *10 Best JavaScript Build Tools and Bundlers Compared*. Online. WPShout. 2024. Dostupné z: https://wpshout.com/best-javascript-build-tools-bundlers/. [cit. 2024-12-20].

MDN WEB DOCS. *HTML: HyperText Markup Language*. Online. Developer Mozilla. 2024. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML. [cit. 2024-12-27].

MDN WEB DOCS. *Performance Monitoring: RUM vs. synthetic monitoring*. Online. Developer Mozilla. 2024. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Performance/Rum-vs-Synthetic. [cit. 2024-11-22].

MICHÁLEK, Martin. *Chrome UX Report: Data o rychlosti webu přímo od uživatelů*. Online. Vzhůru dolů. 2019. Dostupné z: https://www.vzhurudolu.cz/prirucka/chrome-ux-report. [cit. 2024-11-23].

MICHÁLEK, Martin. *Interaction to Next Paint (INP): kladivo na pomalé interakce*. Online. Vzhůru dolů. 2024. Dostupné z: https://www.vzhurudolu.cz/prirucka/metrika-inp. [cit. 2024-11-18].

MICHÁLEK, Martin. *Lazy loading v kontextu webového frontendu: co to je a proč to dělat?* Online. Vzhůru dolů. 2019. Dostupné z: https://www.vzhurudolu.cz/prirucka/lazy-loading. [cit. 2024-12-20].

MICHÁLEK, Martin. *Lighthouse: audit webu od Google*. Online. Vzhůru dolů. 2021. Dostupné z: https://www.vzhurudolu.cz/prirucka/lighthouse. [cit. 2024-12-01].

MICHÁLEK, Martin. *Lighthouse Performance Score (LPS): krásy i pasti metriky všech metrik*. Online. Vzhůru dolů. 2021. Dostupné z: https://www.vzhurudolu.cz/prirucka/metrika-lps. [cit. 2024-11-28].

MICHÁLEK, Martin. *Metrika Total Blocking Time (TBT), zaměřeno na pomalý JavaScript*. Online. Vzhůru dolů. 2021. Dostupné z: https://www.vzhurudolu.cz/prirucka/metrika-tbt. [cit. 2024-11-27].

MICHÁLEK, Martin. *Nástroje pro analýzu rychlosti načtení stránky*. Online. Vzhůru dolů. 2016. Dostupné z: https://www.vzhurudolu.cz/prirucka/rychlost-nastroje. [cit. 2024-11-23].

MICHÁLEK, Martin. *Průvodce formáty obrázků pro web: JPEG, WebP, AVIF, PNG, GIF a SVG*. Online. Vzhůru dolů. 2020. Dostupné z: https://www.vzhurudolu.cz/prirucka/obrazky-formaty. [cit. 2024-12-20].

MICHÁLEK, Martin. *Událost DOM Content Loaded (DCL)*. Online. Vzhůru dolů. 2019. Dostupné z: https://www.vzhurudolu.cz/prirucka/udalost-dcl. [cit. 2024-11-25].

MOREY, Raelene. *Pingdom vs GTmetrix vs WebPagetest: How Are They Different?* Online. WP Rocket. 2023. Dostupné z: https://wp-rocket.me/blog/pingdom-vs-gtmetrix-vs-webpagetest-different/. [cit. 2024-12-09].

OSMANI, Addy a POLLAND, Barry. *Optimize Cumulative Layout Shift*. Online. Web.dev. 2020. Dostupné z: https://web.dev/articles/optimize-cls. [cit. 2024-11-16].

POL, Tushar. *Google Lighthouse: What It Is & How to Use It*. Online. Semrush. 2023. Dostupné z: https://www.semrush.com/blog/google-lighthouse/. [cit. 2024-12-01].

RAYKOVA, Lora. *Core Web Vitals for Ecommerce: How to Increase Online Sales*. Online. NitroPack. 2024. Dostupné z: https://nitropack.io/blog/post/how-to-increase-online-sales-core-web-vitals. [cit. 2024-11-12].

RAYKOVA, Lora. *What is Cumulative Layout Shift (CLS) And How to Fix It*. Online. NitroPack. 2024. Dostupné z: https://nitropack.io/blog/post/fix-cls. [cit. 2024-11-16].

ROSSI, Tata. *Lossless vs Lossy Compression: Actual Difference Explained*. Online. Fix the photo. 2024. Dostupné z: https://fixthephoto.com/lossless-vs-lossy-compression.html. [cit. 2024-12-20].

ROSSI, Tata. *Best Image Optimizer*. Online. Fix the photo. 2024. Dostupné z: https://fixthephoto.com/best-image-optimizer.html. [cit. 2024-12-20].

SHARMA, Vibha. *Purpose of Minification and optimizing Frontend performance*. Online. Medium. 2023. Dostupné z: https://vibhas1892.medium.com/purpose-of-minification-and-optimizing-frontend-performance-758c3e8d9267. [cit. 2024-10].

SHELLHAMMER, Alex a NEEL, Juliette. *The need for mobile speed*. Online. Google Ad Manager. 2016. Dostupné z: https://blog.google/products/admanager/the-need-for-mobile-speed/. [cit. 2024-11-12].

TARUGU, Prudvi. *API Performance Optimization: A Complete Guide to Metrics, Terminology, and Optimization Techniques*. Online. Medium. 2023. Dostupné z: https://medium.com/@prudvi.tarugu/api-performance-optimization-a-complete-guide-to-metrics-terminology-and-optimization-techniques-26f92d0fbfb2. [cit. 2024-12-21].

TODER, Shay. *WHY IS MY LARGEST CONTENTFUL PAINT (LCP) SO MESSED UP?* Online. SpeedSize. 2024. Dostupné z: https://speedsize.com/blog/why-is-my-largest-contentful-paint-lcp-so-messed-up/. [cit. 2024-11-14].

TRAUZOLD, Martin. *Jak používat Google PageSpeed Insights*. Online. Amazon Associates. 2020. Dostupné z: https://amazon-affiliate.eu/cs/jak-pouzivat-google-pagespeed-insights/. [cit. 2024-11-30].

WAGNER, Jeremy. *Interaction to Next Paint (INP)*. Online. Web.dev. 2024. Dostupné z: https://web.dev/articles/inp. [cit. 2024-11-18].

WAGNER, Jeremy a POLLARD, Barry. *Time to First Byte (TTFB)*. Online. Web.dev. 2024. Dostupné z: https://web.dev/articles/ttfb. [cit. 2024-11-25].

WAGNER, Jeremy a POLLARD, Barry. *Optimize Time to First Byte*. Online. Web.dev. 2024. Dostupné z: https://web.dev/articles/optimize-ttfb. [cit. 2024-11-25].

WALTON, Philip. *Web Vitals*. Online. Web.dev. 2020. Dostupné z: https://web.dev/articles/vitals. [cit. 2024-11-12].

WALTON, Philip. *First Contentful Paint (FCP)*. Online. Web.dev. 2023. Dostupné z: https://web.dev/articles/fcp. [cit. 2024-11-27].

WALTON, Philip a POLLARD, Barry. *Largest Contentful Paint (LCP)*. Online. Web.dev. 2020. Dostupné z: https://web.dev/articles/lcp. [cit. 2024-11-14].

WALTON, Philip a POLLAND, Barry. *Optimize Largest Contentful Paint*. Online. Web.dev. 2020. Dostupné z: https://web.dev/articles/optimize-lcp. [cit. 2024-11-14].

WEBGLOBE. *Co to je CDN*. Online. Webglobe. 2024. Dostupné z: https://www.webglobe.cz/poradna/co-je-cdn. [cit. 2024-12-22].

YADAV, Swastik. *Sass: The CSS preprocessor handbook and how does it work?* Online. DEV. 2023. Dostupné z: https://dev.to/swastikyadav/sass-the-css-preprocessor-handbook-and-how-does-it-work-58dn. [cit. 2024-12-27].

Přílohy

Příloha A: CD se zdrojovým kódem aplikace Web Optimizer

Příloha B: Instalace a spuštění aplikace Web Optimizer

Zdrojový kód

Kompletní zdrojový kód projektu je dostupný na GitHubu:
https://github.com/tomasdevs/web-optimizer

Požadavky:

- Node.js verze 18 nebo novější

Instalace a spuštění:

1. **Naklonujte repozitář:**
   `git clone https://github.com/tomasdevs/web-optimizer.git`
   `cd web-optimizer`
2. **Nainstalujte závislosti:**
   `npm install`
3. **Spusťte vývojový server:**
   `npm run dev`
   Aplikace poběží na *http://localhost:5173/* (defaultní Vite port).

Online verze

Aplikace je nasazena a dostupná online: *https://web-optimizer.netlify.app/*