Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Optimalizace webových stránek

Borůvka Martin

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Martin Borůvka

Informatika

Název práce

Optimalizace webových stránek

Název anglicky

Website optimization

Cíle práce

Bakalářská práce je tématicky zaměřena na problematiku optimalizací webových stránek.

Hlavním cílem práce je optimalizace již existující webové stránky a následné zhodnocení pomocí analytického nástroje.

Dílčí cíle bakalářské práce jsou:
- analýza metrik používaných pro měření rychlosti načítání webových stránek,
- vytvoření přehledu dostupných nástrojů pro měření webových stránek,
- vytvoření přehledu častých chyb dělaných při tvorbě webových stránek.

Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Praktická část práce je zaměřena na optimalizaci webových stránek, kde budou uplatněny nejen získané teoretické poznatky, ale i zkušenost autora s danou problematikou. Závěry bakalářské práce budou formulovány na základě teoretických poznatků a výsledků praktické části práce.

Doporučený rozsah práce

40-50 stran

Klíčová slova

optimalizace webu, rychlost webu, web vitals, LCP, FID, CLS

Doporučené zdroje informací

LARSEN, Rob. Beginning HTML and CSS. John Wiley & Sons, Incorporated, 2013. ISBN 9781118340288.
MAURICE, Florence. HTML und CSS Für Dummies: Für Dummies Ser. John Wiley & Sons, Incorporated, 2019. ISBN 9783527821259.
ŘEZÁČ, Jan. Web ostrý jako břitva. Baroque partners, 2014. ISBN 9788087923016.
SHIVAKUMAR, Shailesh Kumar. Modern Web Performance Optimization: Methods, Tools, and Patterns to Speed up Digital Platforms. Apress L. P., 2020. ISBN 9781484265284.
UZAYR, Sufyan bin. Web Performance Optimization: A Practical Approach. Taylor & Francis Group, 2022. ISBN 9781000541342.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Alexandr Vasilenko, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 14. 7. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 27. 10. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 10. 03. 2023

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Optimalizace webových stránek" jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 10. 3. 2023 _____

Poděkování

Rád(a) bych touto cestou poděkoval(a) Ing. Alexandr Vasilenko, Ph.D. za aktivní přístup, cenné rady a pomoc při tvorbě této práce. Dále bych chtěl poděkovat mé manželce, dětem a celé rodině, za veškerou podporu, která se mi z jejich strany dostala.

Optimalizace webových stránek

Abstrakt

Cílem této bakalářské práce je analyzovat rychlost načtení a uživatelský zážitek při načítání na již existující webové stránce, navrhnout potenciální optimalizace a následně je aplikovat.

Pro potřeby této práce byly vybrány webové stránky www.bervia.cz. Tyto webové stránky byly vybrány z toho důvodu, že si je vytvářel přímo majitel webu pomocí frameworků a svých amatérských znalostí. Na této stránce se vyskytovaly základní nedostatky, na kterých bylo možné dobře znázornit základní chyby z pohledu rychlosti načtení při tvorbě webu.

Po analýze webových stránek byly navrženy optimalizaci, které měly největší potenciál na zlepšení hodnot metrik používaných pro vyhodnocení rychlosti načtení. Navržené optimalizace byly aplikovány a zapracovány do zdrojového kódu webových stránek.

Následně byla provedena měření pomocí nástroje Google Lighthouse. Měřeny byly všechny základní metriky, které tento nástroj dokáže vyhodnotit. Měření byla provedena pro 3 základní strany webu: hlavní stranu, stranu portfolia a stranu kontakt.

V poslední části došlo k porovnání hodnot měření před optimalizacemi a po optimalizacích. Výsledky měření byly následně vyhodnoceny a byl navržen další postup.

Klíčová slova: optimalizace webu, rychlost webu, web vitals, LCP, FID, CLS

Website optimization

Abstract

The aim of this bachelor thesis was to analyze the loading speed and user experience when loading on an existing website, suggest potential optimizations and apply the suggested optimizations.

The website www.bervia.cz was selected. This website was chosen because it was created directly by the website owner using frameworks and his amateur knowledge. There were basic flaws in this website, which could be well illustrated in terms of loading speed during web development.

After analyzing the website, optimizations were suggested that had the greatest potential to improve the values of the metrics used to evaluate load speed. The proposed optimizations were applied and incorporated into the source code of the website.

Subsequently, measurements were taken using the Google Lighthouse tool. All the basic metrics that this tool can evaluate were measured. Measurements were taken for 3 main pages of the website: the home page, the portfolio page and the contact page.

In the last part, the measurement values before and after optimizations were compared. The measurement results were then evaluated and a further course of action was proposed.

Keywords: website optimization, page speed, web vitals, LCP, FID, CLS

Obsah

1 Úvod

Webové stránky navštěvuje téměř veškerá lidská populace. Lidé na webových stránkách hledají zábavu, informace, sociální kontakt nebo pomocí nich prezentují sebe či své služby. Slouží taktéž k tvorbě zisku či zprostředkování obchodů.

Uživatelský zážitek z používání webových stránek je z velké části ovlivněn rychlostí načítání a průběhem vykreslování stránky během načítání. Lidé si váží čím dál více svého času, a proto nechtějí čekat na pomalé načtení webové stránky. Z důvodů pomalého načítání mají lidé tendenci stránku opustit a potřebné informace či produkty, kvůli kterému ji vyhledali, nalézt někde jinde.

Rychle načtené webové stránky mají obvykle vyšší konverzní poměr, což znamená, že uživatelé budou s vyšší pravděpodobností provádět akce, jako je: nákup, přihlášení k odběru newsletteru či odeslání kontaktního formuláře.

Rychlost načtení webových stránek má vliv i na SEO. Vyhledávače jako je Google zohledňují rychlost načtení při určování pozic webové stránky při vyhledávání. Stránky, které mají lepší pozice při vyhledávání mají vyšší návštěvnost a může se tím zvýšit i důvěryhodnost webové stránky.

V dnešní době je stále více lidí, kteří používají k procházení webových stránek mobilní zařízení. Rychlé načtení webových stránek na mobilních zařízeních je důležité, protože jejich časté pomalejší připojení k internetu může způsobit významné zpoždění při zobrazení stránek.

Vývojáři velkých e-shopů a webových stránek často dbají na funkčnost a rychlost načtení, ale menší stránky, které se často tvoří pomocí robustních frameworků a šablon zapomínají na důležitost rychlosti načtení a uživatelského zážitku při načítání. Z tohoto důvodu se tato práce zaměřuje na základní chyby, které se vyskytují v menší či větší míře na většině menších webů tvořených amatéry.

2 Cíl práce a metodika

2.1 Cíl práce

Bakalářská práce je tematicky zaměřena na problematiku optimalizací webových stránek. Hlavním cílem práce je optimalizace již existující webové stránky a následné zhodnocení pomocí analytického nástroje.

Dílčí cíle bakalářské práce jsou:

- analýza metrik používaných pro měření rychlosti načítání webových stránek,
- vytvoření přehledu dostupných nástrojů pro měření webových stránek,
- vytvoření přehledu častých chyb vzniklých při tvorbě webových stránek.

2.2 Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Praktická část práce je zaměřena na optimalizaci webových stránek, kde jsou uplatněny nejen získané teoretické poznatky, ale i zkušenost autora s danou problematikou. Závěry bakalářské práce jsou formulovány na základě teoretických poznatků a výsledků praktické části práce.

3 Teoretická východiska

Pro pochopení možnosti optimalizace webových stránek je nejprve potřeba pochopit z čeho jsou stránky složené, a jak jednotlivé vrstvy stránek fungují.

3.1 HTML

HTML je hypertextový značkovací jazyk, který se používá pro vytvoření struktury webových stránek. Poslední verzí jazyka je HTML5. Jazyk HTML5 byl nástupcem předchozích verzí jazyka HTML a zavedl do jazyka nové prvky a možnosti nad rámec předchozí verze HTML. Zároveň také vylepšil nebo odstranil některé stávající funkce (1).

Webové stránky, které jsou vidět v prohlížeči, vypadají tak jak vypadají jen díky kaskádovým stylům (2).

3.2 CSS

Kaskádové styly (CSS) je jazyk, který se používá ke stylování dokumentů a webových stránek psaných ve značkovacím jazyce HTML (3). CSS funguje tak, že umožňuje přiřadit pravidla k prvkům, které se objevují na webové stránce. Tato pravidla určují, jak má být obsah těchto prvků vykreslen. (4)

CSS se řadí mezi základní jazyky otevřeného webu a je standardizován ve všech webových prohlížečích podle specifikace W3C (3).

Tvorba CSS se doporučuje metodou mobile-first. Tato metoda říká, že webové stránky by se měly vytvářet prvotně pro mobilní zařízení, což klade vyšší důraz i na nižší datovou náročnost, a tudíž i rychlejší načtení webu na mobilních zařízeních (5).

3.3 JavaScript

JavaScript (JS) je lightweight skriptovací jazyk běžící na straně klienta. Díky JavaScriptu je možné měnit obsah webové stránky, vytvářet různé dynamické menu, roletky atp.

Díky tomu, že JavaScript pracuje na straně klienta, najde uplatnění například i pro validace webových formulářů. V případě, že do formuláře budou napsány nějaké neplatné hodnoty, tak na to JavaScript upozorní ještě před odesláním formuláře (6).

JavaScript je možné v prohlížeči vypnout, a proto je vždy potřeba myslet na to, aby web fungoval, když uživatel JavaScript zakáže (6).

3.4 Frameworky pro vývoj webových aplikací

Frameworkem bývá označována sada nástrojů, která vývojářům pomáhá v tvorbě webových stránek. Místo toho, aby vývojáři museli programovat weby či aplikace od začátku, tak s frameworkem vývojáři pouze upravují a rozšiřují již připravené komponenty či metody.

Framework často bývá nahráván na web celý, ale využívá se z něho pouze určitá část. Zbytek, který se nevyužívá, ale musí prohlížeč i tak stahovat a tím může docházet k velké datové náročnosti webu a tím i zpomalení načítání (7).

V následující části budou zhodnoceny 2 velmi používané frameworky při tvorbě webových stránek. První je framework Bootstrap a druhý framework jQuery.

3.4.1 Bootstrap

Bootstrap je framework určený pro tvorbu webových stránek. Nabízí vývojářům již předpřipravené komponenty přímo k použití. Výhoda Bootstrapu je taková, že při vývoji webu se nemusí psát veškeré webové prvky od začátku, ale je možné použít předpřipravené komponenty, či je možné je modifikovat podle potřeb. Díky tomu se snižuje časová náročnost vývoje webu (8).

Další výhodou Bootstrapu je, že je open-source, tudíž každý ho může zdarma používat jak k osobním, tak komerčním účelům. Je to nejpoužívanější CSS framework na světe a díky tomu má i obrovskou komunitu lidí, kteří ho používají (8).

Prohlížení webu se čím dál více dostává na mobilní zařízení, a proto je dobře, že je Bootstrap dokonale přizpůsoben pro mobilní zařízení. Jeho kód je psán metodou mobile-first a díky tomu je kompaktnější a podporuje rychlost načtení na mobilních zařízeních (8).

Nejnovější verze je Bootstrap 5.0.2 a minifikovaný CSS balíček má velikost 58,6 kB. Minifikovaný balíček s JavaScriptem má velikost 152 kB (9).

3.4.2 jQuery

Knihovna jQuery je nejrozšířenější JS knihovna na světe. Knihovna má jednoduchou syntaxi a proto lze velmi snadno manipulovat s obsahem na stránce, animacemi či reakcemi na události (9).

Její nesporná výhoda spočívá ve velké dostupnosti různých pluginů. Díky nim je možno do stránky téměř bez práce naimplementovat různé interaktivní obrázkové knihovny, carousely atd. (9)

Stejně tak jako u Bootstrapu je jQuery zdarma, open-source a má velkou komunitu vývojářů (9).

Byť je jQuery datově velmi náročná, obliba využívání je velmi vysoká. Její využití lze již dnes ale obejít i jinými datově méně náročnými frameworky. Její aktuální verze 3.6.2 má po minifikaci velikost 87,8 kB (11).

3.5 Metriky rychlosti

Dlouhodobá životaschopnost každého webu na internetu závisí na jeho schopnosti optimalizovat kvalitu uživatelského zážitku. Web Vitals pomáhá měřit spokojenost uživatelů s webovými stránkami a odhalit oblasti, které je třeba zlepšit (10).

Rychlé načítání stránek souvisí s větší ochotou návštěvníků se tam opakovaně vracet a s větší důvěrou lidí k daným webovým stránkám, V ideálním případě návštěvníci očekávají, že se stránky načtou do 2 sekund. Po třech sekundách je až 40 % uživatelů opustí (11).

Dobrý uživatelský zážitek z webu závisí na řadě faktorů. Společnost Google proto vytvořila program zvaný Web Vitals, který poskytuje hodnocení a doporučení pro webové stránky (10).

V průběhu let společnost Google nabídla řadu nástrojů pro sledování výkonnosti a měření uživatelského zážitku z webových stránek. Zatímco někteří vývojáři umí tyto nástroje používat výborně, pro jiné je obtížné udržet krok s množstvím nástrojů a údajů (10).

V sadě Web Vitals je více metrik. Google vybral ty nejpodstatnější a nazval je jako Core Web Vitals. Tyto metriky by měli měřit všichni majitelé webů. Metriky se taktéž zobrazují ve všech nástrojích od Googlu. Každá metrika z Core Web Vitals představuje jedinečný aspekt uživatelské zkušenosti, je měřitelná, a zachycuje praktické použití klíčového výsledku zaměřeného na uživatele (10).

3.5.1 Core Web Vitals

Postupem času se metriky, které tvoří Core Web Vitals, mění. V roce 2020 se Core Web Vitals soustředil na tři aspekty: (10)

- LCP (největší vykreslení obsahu),
- FID (prodleva prvního vstupu),
- CLS (vizuální stabilita při načítání stránky).

LCP

Metrika LCP (Largest Contentful Paint) počítá čas mezi začátkem načítání stránky a vykreslením největšího obrázku nebo textového bloku viditelného ve výřezu obrazovky monitoru (12).

Stránky by se měly snažit dosáhnout hodnoty 2,5 sekundy nebo méně. V tom případě poskytnou dobrý uživatelský zážitek z pohledu načtení největšího smysluplného obsahu. Výborným měřítkem je 75. percentil načítání stránek rozdělený mezi mobilní a desktopové platformy. Toto pravidlo o 75. percentilu platí pro všechny následující metriky (12).

Prvky, které metrika LCP zahrnuje, jsou uvedeny v rozhraní API Largest Contentful Paint jako: (12)

- <img>,
- <image> uvnitř <svg> elementů,
- <video> (obrázek plakátu),
- elementy s barevným pozadím načteným pomocí funkce url(),
- texty.

FID

FID (First Input Delay) počítá dobu od okamžiku, kdy uživatel poprvé vstoupí do interakce se stránkou (tj. když klikne na odkaz, klepne na tlačítko nebo použije vlastní ovládací prvek s podporou JavaScriptu), do okamžiku, kdy je prohlížeč skutečně schopen začít zpracovávat obsluhy událostí v reakci na tuto interakci. Metrika FID tedy pomáhá měřit první dojem uživatele z interaktivity a odezvy webu (13).

Aby stránka poskytovala dobrý uživatelský zážitek, měly by se vývojáři snažit mít na stránkách zpoždění prvního vstupu 100 milisekund nebo méně (13).

CLS

Martin Michálek říká, že CLS (Cumulative Layout Shift) je *„Metrika, která udává stabilitu vzhledu stránky během vykreslování.“* (14). K posunu rozvržení stránky dojde vždy, když viditelný prvek změní svou pozici z jednoho vykresleného místa na jiné (15).

K zaznamenání změny rozvržení dochází tehdy, když se jeden nebo více jednotlivých posunů rozvržení objeví v rychlém sledu za sebou s intervalem kratším než 1 sekunda mezi jednotlivými posuny a s celkovou dobou trvání okna maximálně 5 sekund (15).

Vývojáři by se měly snažit mít na webových stránkách skóre CLS 0,1 nebo méně (15).

Web Vitals

V následující kapitole budou představeny metriky, kterou nejsou součástí Core Web Vitals, ale Google pomocí nich taktéž hodnotí výkon webových stránek.

TTFB

TTFB (Time to First Byte) je metrika, která vzniká ve chvíli, kdy se stáhne první bajt ze stránky. Metrika tedy ukazuje rychlost serveru a backendové části aplikace (16).

TTFB je součtem následujících fází žádosti: (17)

- doba přesměrování,
- doba spuštění service workeru,
- vyhledávání DNS,
- připojení a jednání s TLS,
- doba trvání požadavku až do okamžiku, kdy dorazil první bajt odpovědi.

Dle Michálka by se měla metrika držet co nejvíce pod půl vteřinou, a to i na pomalých zařízeních (18).

FCP

Metrika First Contentful Paint (FCP) počítá dobu od začátku načítání stránky do okamžiku, kdy je na obrazovce vykreslena jakákoli část obsahu stránky. Tato metrika interaguje s texty, obrázky (včetně obrázků vložených pomocí css atributu background-image), prvky <svg> nebo jinými než bílými prvky <canvas> (19).

Aby stránky poskytovaly dobrý uživatelský zážitek, měly by se vývojáři snažit o to, aby první vykreslený obsah byl do 1,8 sekund (19).

TTI

TTI (Time to interactive) počítá dobu od zahájení načítání stránky do načtení jejích hlavních dílčích částí, kdy je stránka schopna spolehlivě a rychle reagovat na vstupy uživatele (21).

Aby stránky poskytovaly dobrý uživatelský zážitek, měly by se vývojáři při testování snažit, aby doba interaktivity byla kratší než 5 sekund (21).

TBT

Metrika TBT (Total Blocking Time) počítá celkovou dobu mezi FCP a TTI. Je to tedy čas, kdy stránka není schopna reagovat na uživatele (20).

Stránka je považována za "zablokovanou" vždy, když se objeví „dlouhá úloha“. To je úloha, která trvá déle než 50 milisekund. V tom případě říkáme, že je hlavní vlákno "zablokováno", protože prohlížeč nemůže přerušit probíhající úlohu. V případě, že uživatel provede interakci se stránkou uprostřed dlouhé úlohy, tak musí prohlížeč počkat na dokončení úlohy a až poté je schopný reagovat (20).

Pokud je úloha dostatečně dlouhá (cokoli nad 50 ms), je pravděpodobné, že si uživatel zpoždění všimne a stránku bude vnímat jako pomalou nebo trhavou (20).

Aby stránky poskytovaly dobrý uživatelský zážitek, tak by vývojáři měly při testování usilovat o TBT metriku kratší než 200 ms (20).

3.6 Nástroje pro měření rychlosti

V předchozí části byly popsány metriky, pomocí kterých lze hodnotit výkon webu. Nyní budou představeny nástroje, které dané metriky hodnotí a pomáhají vývojářům optimalizovat webové stránky. Nástrojů pro měření je více, v této práci budou představeny pouze ty nejpoužívanější.

3.6.1 PageSpeed Insights

Služba PageSpeed Insights (PSI) měří výkon webových stránek pro počítače a mobilní zařízení. Zároveň uživateli nabízí doporučení, jak stránku vylepšit (22).

Služby PSI nabízí laboratorní data, ale i data od reálných uživatelů. Laboratorní data jsou vhodná pro zjištění problémů s výkonem, jelikož jsou shromažďována v kontrolovaném prostředí a jsou denně aktualizována. Ne vždy však dokážou zachytit problémy reálných uživatelů. Data od reálných uživatelů mají omezenější sadu metrik, za to však zobrazují výsledky od uživatelů, kteří stránky aktivně využívají (22).

Data o reálných uživatelích jsou založena na datové sadě Chrome User Report (CrUX). PSI je uvádí pouze u metrik FCP, FID, LCP a CLS. PSI uvádí taktéž od reálných uživatelů experimentální metriky INP a TTFB. Data od nich jsou sbírána a vypočítána vždy za posledních 28 dní (22).

Aby bylo možné zobrazit data o reálných uživatelích, tak musí mít stránka dostatečnou návštěvnost, jinak nebude mít dostatek dat pro CrUX. Stránky tedy nemusí mít

dostatek dat, pokud byly teprve nedávno zveřejněny, nebo mají nedostatečnou pravidelnou návštěvnost. V tomto případě PSI zprůměruje výsledky ze všech stránek webu. Pokud by se stalo, že nestačí ani zprůměrovat výsledky ze všech stránek, tak PSI nebude schopen zobrazit žádná data od reálných uživatelů (22).

PSI hodnotí metriky do tří skupin.

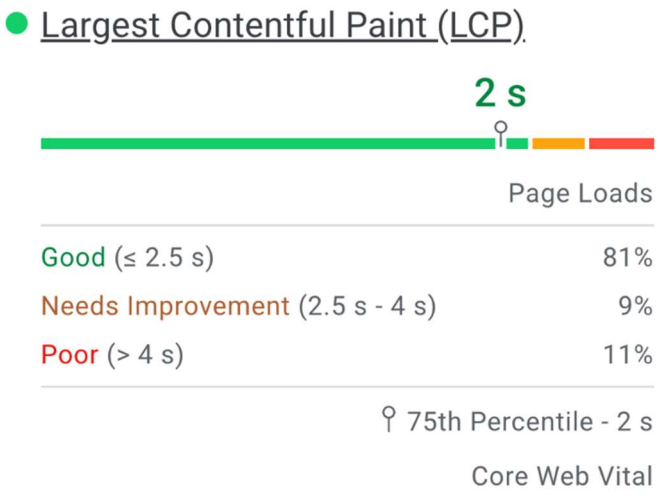
- Dobré
- Potřebuje zlepšit
- Špatné

V souladu s Web Vitals stanovuje PSI prahové hodnoty jednotlivých metrik.

	Dobré	Potřebuje zlepšit	Špatné
FCP	[0, 1800ms]	(1800ms, 3000ms]	Více než 3000ms
FID	[0, 100ms]	(100ms, 300ms]	Více než 300ms
LCP	[0, 2500ms]	(2500ms, 4000ms]	Více než 4000ms
CLS	[0, 0.1]	(0.1, 0.25]	Více než 0.25
INP (experimentální)	[0, 200ms]	(200ms, 500ms]	Více než 500ms
TTFB (experimentální	[0, 800ms]	(800ms, 1800ms]	Více než 1800ms

Tabulka 1: Metriky a jejich hodnocení

PSI prezentuje rozdělení metrik, aby vývojáři zjistili, jaký počet uživatelů má jaké zkušenosti s danou stránkou či celým webem. Rozdělení do skupin je reprezentováno pomocí barev. Dobré je zelená, potřebuje zlepšit žlutá a špatné červená (22).



Obrázek 1 Zobrazení metriky LCP ve službě PageSpeed Insights (23)

U obrázku číslo 1 lze vidět, že 81 % uživatelů se zobrazí největší obsah do 2,5 s, přičemž 75. percentil je 2 vteřiny. Do druhé skupiny spadá 9 % uživatelů, kterým se metrika CLS aktivuje mezi 2,5 a 4 vteřinami. PSI vyhodnotil stránku špatně pro 11 % uživatelů. Těm se největší obsah načítá déle než 4 vteřiny (22).

Výkonnostní skóre

V horní části se zobrazuje PSI skóre. To zahrnuje simulovaný výkon stránky. Toto skóre je určeno na základě vyhodnocení stránky v nástroji Lighthouse, který shromažďuje a analyzuje diagnostická data o stránce (22).

PSI výkonnostní skóre se taktéž dělí do 3 skupin:

- dobré – vyšší než 90 bodů,
- potřebuje zlepšit – 50 až 90 bodů,
- špatné – méně než 50 bodů.

3.6.2 Chrome UX report

CrUX neboli Chrome UX Report sbírá data (zejména Core Web Vitals) o kvalitě uživatelského prožitku na webových stránkách a ukládá si je do databáze BigQuery. Tyto data jsou všem volně dostupná. Z dat Chrome UX report lze vyčíst údaje o reálné rychlosti webu u uživatelů (24).

Oproti výše zmíněným nástrojům je nevýhodou, že data z Chrome UX Reportu jsou dostupná obvykle až po několika týdnech a neobsahují všechny potřebné metriky. Data navíc pochází pouze od uživatelů prohlížeče Chrome (24).

Velkou výhodou Chrome UX Report je, že Google poskytuje tyto data zdarma. Obdobná komerční řešení jako LUX, či SpeedCurve jsou velmi drahá (24).

Data z CrUX se taktéž vyplatí sledovat z důvodu SEO. Má se za to, že je Google využívá pro vyhodnocování rychlosti indexovaných webů v prohlížeči, což je jeden z mnoha aspektů řazení výsledků vyhledávání (24).

Metriky uložené v Chrome UX Report

Není zde tolik metrik jako u syntetických měření, ale autoři postupem času přidávají stále další: (24).

- first Paint (vykreslení čehokoliv),
- first Contentful Paint (první vykreslení smysluplného obsahu),
- largest Contentful Paint (vykreslení největší části obsahu),
- DOM Content Loaded (rozparsování HTML prohlížečem),
- load (stažení celé stránky).

Metody k získávání Dat z Chrome UX Report

Z této užitečné databáze lze získávat data více způsoby. Mezi čtyři základní se řadí následující (24).

- PageSpeed Insights
    - Data se sbírají a počítají za posledních 28 dní, takže jsou velmi aktuální.
    - Dostupné jsou pouze metriky LCP a FID.
- Google Search Console
    - Služba nabízí webmasterům informace o tom, jak si web vede. Nově experimentálně zobrazuje data z Chrome UX Report pomocí grafů.
    - Data se ukazují s čerstvostí na dny, ale služba je stále v experimentálním režimu. Data úplně neodpovídají tomu, co je vidět v PageSpeed Insights a dalších nástrojích.
- CrUX run
    - Služba nenabízí čerstvá data, ale upřednostňuje pohled na historii vývoje rychlosti po měsících. Tyto data jsou vhodná k vyhodnocování v dlouhém období.

- Google Data Studio
    - Nástroj slouží k vizualizaci dat a tvorbě živých dashboardů.
    - K nástroji se připojí data, nastaví se způsob vykreslení dat a nasdílí se lidem, kteří mají grafy sledovat.

3.6.3 Google Lighthouse

Lighthouse je velmi důležitý nástroj pro analýzu technické kvality webu. Jeho hlavní využití je k analýze rychlosti načítání, ale pokrývá i další oblasti pro zlepšování webových stránek. Lighthouse dokáže objevit problémy jak na úrovni designu, tak frontendového kódu. Po vyhodnocení dokáže poradit, jak zjištěné problémy vyřešit (25).

Lighthouse je open-source a je možné ho spustit na jakékoli webové stránce. Měření je možné spustit z Chrome DevTools, příkazové řádky, nebo jako modul Node. Lighthouse dostane url adresu, kterou změří, a proběhne série auditů. Neúspěšné audity lze považovat za indikátory, jak stránku lze vylepšit (28).

Mezi jeho výhody patří snadná dostupnost pro všechny, rychlé výstupy a rozumné rady. Naopak mezi nevýhody by se dalo zařadit to, že dává spíše základní přehled o webu, výsledky mohou být ovlivněny výkonem počítače, na kterém se testuje, a dělá pouze syntetickou analýzu, tudíž nedává obrázek o celé šíři problémů (25).

Lighthouse měří a vyhodnocuje data v těchto oblastech.

- Performance – rychlost načítání.
- Progressive Web App – jak si web vede v oblasti progresivních webových aplikací.
- Best Practices – osvědčené postupy k tvorbě webů, například k bezpečnosti či používání zastaralých technologií.
- Accessibility – přístupnost webu.
- SEO – jak je web připraven na indexování vyhledávači.

V případě měření rychlosti načítání Lighthouse poskytuje data o metrikách:

- Performance,
- FCP (First Contentful Paint),
- SI (Speed Index),
- LCP (Largest Pontentful Paint),

- TTI (Time To Interactive),
- TBT (Total Blocking Time),
- CLS (Cumulative Layout Shift).

Cílem Lighthousu je poskytovat vývojářům rady, které jsou relevantní a použitelné pro jejich webové stránky. Za tímto účelem poskytuje dvě funkce, které umožňují přizpůsobit Lighthouse k potřebám uživatelů (28).

Stack packs

Vývojáři používají pro vývoj webových stránek obvykle mnoho různých technologií pro backend webové aplikace, CMS, JavaScriptové frameworky atp. Stack packs umožňuje zjistit, na jaké platformě je web postaven a podle toho je Lighthouse schopen poskytovat relevantnější a použitelnější rady namísto obecných doporučení (28).

Lighthouse pluginy

Díky Lighthouse pluginům si můžou vývojáři přizpůsobit nástroj pro jejich specifické potřeby. Vývojáři si mohou sami vytvořit plugin a implementovat si ho do Lighthousu podle jejich potřeb. Zároveň lze využívat i pluginy ostatních vývojářů.

3.7 Tipy pro zrychlení webu

V následující sekci budou vypsány tipy pro zrychlení webů, které jsou se často na webových stránkách vyskytují.

3.7.1 Protokol HTTP

Hypertext Transfer Protocol (HTTP) je protokol pro přenos hypertextových dokumentů jako je například již zmíněné HTML. Zajišťuje komunikace mezi webovým prohlížečem a serverem. Protokol se řídí modelem klient-server, kdy klient otevře spojení, zadá požadavek a čeká, než obdrží odpověď od serveru (26).

Protokol byl navržený počátkem 90. let a postupně se vyvíjel. Jedná se o protokol aplikační vrstvy, který je odesílán přes TCP nebo šifrované TLS spojení. Aktuálně se používají 3 verze protokolu: HTTP/1.1, HTTP/2 a HTTP/3. (27).

HTTP skládá kompletní HTML dokument z různě načtených dílčích částí jako je text, popis rozvržení, obrázky, videa a další. Zprávy zaslané klientem (obvykle webovým prohlížečem) se nazývají požadavky a zprávy zaslané serverem jako odpověď se nazývají odpovědi (27).

HTTP/2

HTTP/2 je verze protokolu, která pomáhá urychlit načítání webů. Protokol je podporován všemi moderními prohlížeči a webovými servery (28).

Ve verzi HTTP/1.1 bylo potřeba slučovat více vývojářských souborů do jednoho, jelikož jednotlivé požadavky původní protokol zpracovával příliš pomalu. Díky HTTP/2 lze soubory více rozdrobit a načítat je jen tam, kde jsou potřeba. Toto vylepšení je díky tomu, že je tento protokol nově binární a rychleji parsuje a přenáší data. Zároveň zvládá oproti původnímu protokolu lépe komprimaci dat, podporuje multiplexing (v jednu chvíli lze po síti poslat více požadavků) a umí prioritizaci (lze definovat prohlížeči, jaký obsah má stáhnout jako první) (28).

V případě, že je web plně optimalizovaný pro HTTP/1.1, tak může přechod na novější protokol znamenat dokonce zpomalení webu. Proto je tedy potřeba při přechodu změřit výkon webu a případně připravit web pro HTTP/2 (28).

HTTP/3

Třetí verze protokolu je ještě stále ve vývoji a její podpora zatím není velká (29). Oproti druhé verzi je největší změnou nahrazení protokolu TCP za protokol QUIC. HTTP/2 má v sobě zabudovaný multiplexing, což umožňuje odesílat více zdrojů současně přes jedno připojení. Protokol TCP bohužel tuto výhodu zpomaloval, jelikož jeho pakety jsou přijímány v takovém pořadí, v jakém byly odeslány. Pokud u protokolu TCP dojde ke ztrátě paketu, pak je celé připojení blokováno, dokud není paket přijat (29).

Protokol QUIC toto řeší tím, že mezi body A a B vytvoří několik multiplexovaných spojení, která se navzájem nemohou blokovat a díky tomu nevznikají problémy jako u protokolu TCP (29).

HTTP/3 pravděpodobně nebude podporovat prioritizaci, jelikož v druhé verzi protokolu s ním byly často potíže, kvůli špatnému nastavení serverů a prohlížečů. Každý prohlížeč k prioritizace přistupuje jinak. Safari se obvykle snaží stáhnout všechno najednou, zatímco Firefox a Chrome první stahuje důležité CSS nebo blokující JS (29).

3.7.2 Lazyloading

Lazyloading na webu označuje techniku, která zajišťuje, že se obsah stránky načte až ve chvíli, kdy ho uživatel potřebuje vidět. Prohlížeč zajistí, že stáhne pouze prvky ve viditelné částí obrazovky a další bude stahovat až v případě, že k nim uživatel po stránce posune, zobrazí je kliknutím, či najetím myši. Nasazením lazyloading na web lze na jedné stránce ušetřit až několik megabajtů dat (30).

Ve webdesignu se nejčastěji lazyloadují:

- obrázky,
- videa,
- iframy,
- složité DOM struktury.

Díky tomu, že prohlížeč hned při načtení nemusí stahovat veškerý obsah stránky, tak se stránka obvykle načte rychleji a uživatelům z mobilních připojení nevyčerpá veškerá data (FUP) (30).

Google doporučuje, aby strom DOM měl pod 1500 uzlů. Velký model DOM zpomaluje veškeré operace nad stromem. Mezi takové operace patří například

překreslování stránky atp. Díky lazyloadingu můžeme uzly stromu načítat až když je bude uživatel potřebovat. Může se jednat o velká menu nebo například schované záložky v navigaci čekající na akci uživatele (30).

Obrázky vložené v elementu <img> mají většinou přednost ve stahování před linkovaným CSS. Z důvodu lazyloadingu můžeme nepotřebné obrázky stahovat až v případě potřeby a proto se rychleji stáhnout zdroje pro první vykreslení stránky jako jsou CSS, fonty, ikony, JS (30).

Obrázek 2: Ukázka fungování lazyloadingu

3.7.3 Preload

Meta značka preload určuje prioritu pro stažení prvků stránky a v případě JS odděluje stažení od spuštění. Preload by se měl používat pro prvky, které budou zobrazeny na stránce ihned po jejím načtení (31). Preload je podporován všemi moderními prohlížeči (32).

Preload je potřeba používat pouze v případě, že vývojář zná techniku použití meta značky, jelikož narušení přirozeného vodopádu stahování stránky může způsobit naopak zpomalení načítání webu (31).

Co se týče podpory prohlížečů, tak podle webu www.caniuse.cz podporují tuto meta značku všechny moderní webové prohlížeče (31).

Obrázek 3 Podpora meta značky preload podle služby CanIUse (33)

Meta značka preload obsahuje nepovinný atribut „as“. Atribut by se měl přidávat vždy, abychom prohlížeči pomohli určit o jaký typ prvku se jedná. Jednotlivé typy prvků mají každý jinou prioritu. Typ „style“ má prioritu nejvyšší, naopak typ „script“ má prioritu nízkou nebo střední (31).

Zde je výčet nejpoužívanějších.

Hodnota	**Typ souboru**
Style	CSS soubor
Script	Soubor s JavaScriptem
Font	Soubor s fontem, např WOFF2
Image	Obrázek
Audio	Audio, typicky v prvku <audio>
Video	Video soubor, typicky v prvku <video>

Tabulka 2: Výčet nejpoužívanějších typů souborů pro preload

Další nepovinným atributem je „Type“. Type předá prohlížeči informaci, o jaký typ prvku se jedná a prohlížeč se rozhodne, zda tento typ podporuje. Pokud daný typ prvku prohlížeč nepodporuje, tak ho ani nebude stahovat. Například typ video/webm načtou

všechny prohlížeče, které ho podporují. Stáhnou ho tedy všechny moderní prohlížeče až na Safari (31).

3.7.4 CSS

CSS je blokující zdroj vykreslení, což znamená, že aby se vykreslila stránka, tak se musí nejdříve stáhnout CSS. Aby se stránka načetla rychle, tak musí být i CSS soubor co nejmenší. Každý kilobajt se počítá, a proto by se měla optimalizaci CSS věnovat značná pozornost při tvorbě webu (34).

Podle Google Lighthouse by metrika First Contentful Paint (FCP) měla být do 1,3 s. To by znamenalo, že při rychlosti Fast 3G by muselo pro HTML a CSS stahovat pouze 14 kB dat. Datová náročnost 14 kB je opravdu málo a většina českých e-shopů a webů se do této hodnoty nevejde (34).

Nyní budou představeny jednotlivé možnosti k optimalizaci CSS.

Mazání CSS

Obvykle se projekty vyvíjejí postupně a díky tomu velikost kódu narůstá. Přidává se nový kód, ale starý nepoužívaný se často nemaže. Vývojáři většinou starý kód nemažou, protože mají obavu, že dojde k chybám (34).

Jedna možnost ke zjištění nepoužívaného kódu je otevřít si v Chrome DevTools záložku Coverage. Tam lze zjistit jaký kód se stránce používá a jaký ne. Díky tomu můžeme získat lepší přehled o efektivním využívání našeho kódu. Nicméně to, že se kód na stránce aktuálně nepoužívá neznamená, že ho lze smazat. Aktuálně nevyužívaný kód se může používat například při responzivitě dané stránky nebo po zobrazení nějakého prvku po interakci uživatele. Z tohoto důvodu je potřeba si vždy ověřit, zda daný kus kódu může být smazán či ne (34).

Další metodou zjištění a odstranění nepoužívaného kódu jsou aplikace jako DropCSS nebo UnCSS, nicméně jejich účinnost taktéž není moc významná (34).

Nejlepší a nejpracnější metodou k odstranění starého nepoužívaného kódu je refactoring. U refactoringu jde o to projít postupně veškerý kód, najít nepoužívané části a ty smazat, či používané části kódu přepsat tak, aby byly efektivnější. Refactoring je časově velmi náročný, ale u velkých projektů se vývojáři bez této metody neobejdou, pokud chtějí, aby měl web dobré výsledky (34).

Upravení cache a její invalidace

V souboru .htaccess nebo podobném serverovém konfiguračním souboru je potřeba nastavit instrukce pro dlouhodobé ukládání souborů například na 1 rok. Po této úpravě budou CSS soubory uložené v mezipaměti prohlížeče po dobu jednoho roku, pokud daný CSS soubor nebude změněn (34).

Je nutné, aby nasazovací proces uměl verzovat CSS soubory kvůli invalidaci cache. Soubory můžou mít například za názvem v1, v2, v3 atp. Díky tomu zůstanou uživateli v cachi načtené CSS soubory z minulé návštěvy webu a stáhnout se pouze ty, které byly vývojářem modifikovány (34).

CSS dělení podle šablon

Není efektivní mít veškerý CSS kód v jednom souboru a při příchodu na jednu určitou stránku stahovat kód i pro všechny ostatní stránky. Při příchodu na homepage je pro vykreslení obsahu relevantní pouze kód, který se používá pro homepage, a proto není potřeba, aby prohlížeč v tuto chvíli stahovat i kód používaný například v detailu produktu (34).

Z tohoto důvodu je výhodné jeden velký CSS soubor rozdělit na více menších a ty načítat pouze na stránkách, kde je to potřeba. Tím může vzniknout speciální soubor pro homepage, výpis produktů, detail produktu, checkout atd. (34).

Dělení CSS podle typu

V případě, že je na webu HTTP/2 a správné nastavení cache, tak se soubory mohou dělit nejen podle šablon, ale také podle typu. Vytvoří se speciální CSS soubory base.css, libs.css, helpers.css atd (34).

Toto rozdělení je výhodné kvůli tomu, že uživatel nemusí vždy stahovat velký objem dat i když se změní jen malá část. V případě spojení všech těchto základních CSS do jednoho souboru, tak i kdyby se změnil jeden řádek, tak bude muset uživatel stáhnout celý balík CSS (34).

Dělení CSS podle komponent

Soubory s komponentami jsou zpravidla největší, jelikož obsahují největší část designu webu. V ideálním případě je potřeba rozdělit soubor na jednotlivé komponenty a ty načítat pouze na stránkách, kde se daná komponenta používá (34).

3.7.5 Způsoby vložení JavaScriptu do stránky

JavaScript lze vložit do stránky několika způsoby. Z pohledu rychlosti načítání webu je úplně nejhorší ho vložit do elementu <head> a nedat mu žádné nastavení (36).

Bez nastavení

V případě tohoto použití musí prohlížeč při nalezení JS přestat parsovat HTML, stáhnout daný JS soubor a spustit ho. Až poté může pokračovat v parsování HTML. Prohlížeč musí čekat, jelikož by v externích skriptech mohl být kód, který by mohl ovlivnit strukturu DOM stránky (36).

Tento způsob načtení JS obvykle zhoršuje metriky First Paint (FP) a First Contentful Paint (FCP) (36).

```
<header class="main-banner">
  <script src="script1.js"></script>
</header>
```

Obrázek 4: Ukázka vložení skriptu bez nastavení

Defer

V případě použití defer se script stáhne souběžně při parsování stránky, ale je vyhodnocen až když parsování HTML skončí. V tomto případě je zaručené pořadí (36).

```
<header class="main-banner">
  <script src="script1.js" defer></script>
</header>
```

Obrázek 5: Ukázka vložení skriptu s nastavením defer

Async

Jedná se o asynchronní spouštění. Prohlížeč stáhne skript při parsování, vyhodnotí ho hned po stažení, tedy potencionálně před dokončením parsování. Nicméně zde není zaručené pořadí (36).

```
<header class="main-banner">
  <script src="script1.js" async></script>
</header>
```

Obrázek 6: Ukázka vložení skriptu s nastavením async

JavaScriptový modul

Skripty se společně s jeho závislostmi stáhnou souběžně při parsování. Jsou vyhodnoceny po ukončení parsování a jejich pořadí je garantováno. Modul je ve výchozím stavu nastaven jako defer, tudíž v tomto případě není potřeba uvádět tento atribut (36).

```
<header class="main-banner">
  <script src="script1.js" type="module"></script>
</header>
```

Obrázek 7: Ukázka vložení skriptu s nastavením module

3.7.6 Kritické styly

Kritické styly pomáhají ke zlepšení rychlosti zobrazení stránky, ale na rychlost načtení celé stránky téměř nemají vliv. Jde o vkládání nezbytného CSS kódu přímo do HTML kódu v inline podobě (35).

Prohlížeč vždy čeká na kompletní načtení CSS a v tomto čase je na stránce zobrazeno pouze bílo. Díky kritickým CSS se v tomto čase na stránce zobrazí alespoň základní obsah a uživatel nemusí čekat na kompletní stažení CSS souboru, aby něco viděl (35).

Kritické CSS se využívají hlavně pro první návštěvu uživatele na stránce. Jakmile uživatel přejde na jinou stránku, nebo web navštíví opětovně, tak již má CSS uložené v cache a díky cookies by se mu kritické CSS již neměly zobrazovat (35).

3.7.7 Vhodné využití obrázkových formátů na webu

Obrázky jsou důležitým prvkem pro design webu. Nevýhodou obrázků je, že mají většinou velkou datovou náročnost. Proto se v této části představí, jaké formáty obrázků by se měly, v jakých případech na webu používat (37).

Fotky

Na fotky je vhodné používat JPEG. Je to formát vhodný pro ukládání fotografií a podobného rastrového obsahu. Fotkám vládne už desetiletí a podporují ho všechny moderní prohlížeče. JPEG má omezení na 8bitové snímky a zároveň nemá podporu pro alfa kanál (průhlednost). Formát má jen ztrátovou kompresi (37).

Nové formáty jako je WebP nebo AVIF zvládají komprimaci a kvalitu fotek lépe než JPEG a podpora WebP u prohlížečů je velmi častá, ale i tak je obvykle nutné mít fotografii v JPEG jako alternativní řešení pro prohlížeče, které WebP nepodporují (37).

Bannery s textem nebo obrázky s průhlednosti

Pro bannery, kde se používá v obrázku text, nebo obrázky, kde je použita průhlednost se nejvíce hodí formát PNG. Je to formát s bezeztrátovou kompresí a zvládá průhlednost obrázku neboli takzvaný alfa kanál. (37).

Animace

Pro animace na webu se v minulosti hojně používal formát GIF. Formát používá bezeztrátovou kompresi a je omezen na maximálně 256 barev (37).

V současnosti se již GIF zdá být zbytečný. Lze ho nahradit pomocí HTML5 videa. Animace je možné dělat také v CSS, JS nebo pomocí formátu SVG (37).

SVG je zkratka pro Scalable Vector Graphics. Jedná se o vektorový formát. V současnosti se ve webovém prostředí většinou používá pro ikony, infografiky či animace.

U animací se nedá určit jeden vhodný formát pro použití. Záleží na mnoha faktorech a vždy je potřeba zvolit ten správný podle způsobu použití (37).

3.7.8 Font deskriptor font-display

Tato vlastnost se využívá pro určení, jak se bude vykreslovat webový font během stahování. Deskriptor se používá v pravidle font-face (38).

```
@font face {
  src: url(roboto.woff2) format("woff");
  font-family: "Roboto";
  font-display: fallback;
}
```

Obrázek 8: Ukázka vykreslení fontu pomocí font-face s deskriptorem font-dsisplay

Životní cyklus zobrazení fontu má tři fáze (38).

1. Interval blokování
    - Během této části je obsah schovaný a čeká se na stažení fontu. Po stažení se font zobrazí.
2. Interval náhrady
    - V této části se vykresluje náhradní (systémové písmo) a čeká se na stažení správného fontu. Poté, pokud se se prohlížeči povede font stáhnout, tak se provede náhrada.
3. Interval selhání
    - Pokud se v předchozích dvou fázích nepovedlo stáhnout font, tak zůstane viditelný fallback, neboli náhradní řešení. Většinou se jedná o systémové písmo.

Fallback

Hodnota fallback zajišťuje, že prohlížeč po stažení CSS bude čekat jednu desetinu sekundy na stažení zvoleného webfontu. Pokud ho prohlížeč stihne stáhnout, tak rovnou vykreslí text s tímto písmem. Pokud ho nestihne stáhnout do desetiny sekundy, tak text vykreslí v systémovém písmu, a zvolený webfont se vyrenderuje až po stažení všech potřebných souborů (38).

Block

Tato hodnota dá prohlížeči třísekundový interval pro blokování a poté nastane nekonečný interval náhrady. Podle této hodnoty se aktuálně chová většina moderních prohlížečů (38).

Toto nastavení je vhodné k vykreslení kratších textů, kdy vývojáři záleží na tom, aby nenastalo probliknutí mezi systémovém fontem a správným fontem. Typicky by se mohlo jednat o hlavní nadpis na stránce, kde by byl systémový font výrazně odlišný od správného fontu a došlo by k velké změně (38).

Swap

Hodnota swap zajistí krátký čas blokování (100ms nebo méně) a nekonečný čas náhrady. V praxi to znamená, že nějaké písmo je vidět vždy kromě první desetiny sekundy. Nejdříve se zobrazí písmo systémové a poté zvolený webfont. Nevýhodou může být, že i na rychlých připojeních dochází k probliknutí mezi písmy (38).

Tato hodnota by se měla používat na kratší texty, kde je důležité sdělení, ale mezi fonty není velký typografický rozdíl. Může se jednat o nadpisy článků, názvy produktů atp. (38).

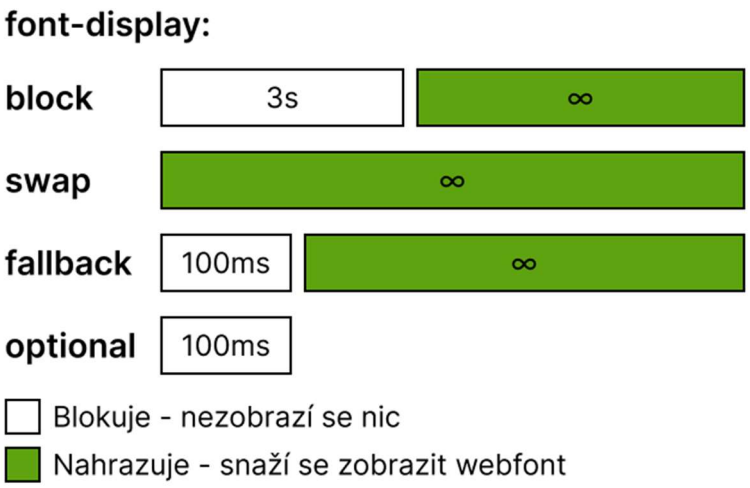
Optional

Hodnota optional nařídí prohlížeči velmi krátký blokující interval (obvykle 100ms) a systém náhrady není žádný (38).

Toto nastavení je vhodné pro delší texty. Na rozdíl od hodnoty fallback dává prohlížeč přednost stabilitě obsahu. Tudíž pokud prohlížeč nestihne font stáhnout během blokujícího intervalu, tak zůstane vidět text se systémovým fontem a nepřekreslí se (38).

Auto

V tomto případě má prohlížeč právo rozhodnout o tom, jak se font vykreslí. Většina výchozího chování prohlížečů odpovídá hodnotě block (38).



Obrázek 9: Životní cyklus zobrazení fontu

4 Vlastní práce

Pro praktickou část bakalářské práce budou použité webové stránky české firmy Bervia (https://www.bervia.cz). Stránky budou optimalizovány z pohledu rychlosti a uživatelské přívětivosti při načítání. Úpravy budou zapracovávány na základě poznatků z teoretické části.

Budou měřeny tři stránky webu (hlavní strana, strana portfolio a strana kontakt). Každá z těchto stránek webu bude měřena desetkrát a hodnoty budou následně zprůměrovány. Měření proběhnou pomocí nástroje Google Lighthouse v anonymním okně v prohlížeči Google Chrome.

Na základě doporučení nástroje Google Lightouse a poznatků z teoretické části bude vytvořen seznam možných optimalizací. Optimalizace budou následně zapracována a aplikována na nový web.

V poslední fázi bude nový web změřen stejným postupem jako web starý a výsledky měření budou porovnány.

4.1 Firma Bervia

Hlavní činností firmy Bervia je natáčení a následná postprodukce videí. Firmu založil pan Michal Bergmann v roce 2018. Web je využíván pro prezentaci jejich práce, portfolia a slouží jako místo, kde klienti naleznou veškeré důležité informace.

Do portfolia firmy spadá natáčení firemních videí, reklamních videí, videí z akcí, aftermovies či animace. Uživatelé si na webu mohou videa a popis prohlédnout na jednotlivých stránkách webu.

4.2 Analýza webu

Web je laděn do modré barvy s bílými prvky a je složen z deseti samostatných stránek, na které se lze prokliknout z hlavní navigace, navigace v patičce, či z odkazů na jednotlivých stránkách.

Na webu se používají externí knihovny Bootstrap 4, jQuery, owlCarousel, typed.js a wow.js. Web je responzivní, a tedy snadný k používání na mobilních zařízeních.

Hlavní strana

Hlavní strana je tvořena navigací, která je zafixovaná v horní části webu, tudíž i v případě posuvu po stránce je navigace stále vidět. Navigace je totožná na všech stránkách webu. Následuje hlavní banner, kde pomocí knihovny typed.js mění text a vytváří dojem, jako by ho někdo psal na klávesnici. Pod hlavním bannerem je pomocí knihovny owlCarousel vytvořen kolotoč obrázků s logy klientů. Klikem na jednotlivá loga se lze dostat na webové stránky klientů. Další v pořadí je krátký popis tvorby videa s rozcestníkem do portfolia Následuje sekce s odkazy na stránku kontakty, proces tvorby a případová studie. Ve spodní části webu je již pouze drobná infografika k důležitosti tvorby videa pro firmy a sekce s popisem koho firma podporuje. Na konci stránky je patička s kontaktními informacemi a rozcestníkem vedoucím na stránky portfolia. Patička je stejně tak jako navigace totožná na všech stránkách webu.

Podstránky portfolia

Každá podstránka, tedy i stránka portfolia má v horní části banner vytvořený pomocí owlCarouselu s titulkem a krátkou anotací k dané stránce. Pod bannerem obvykle následuje výpis videí nahraných na YouTube a vložených do stránky pomocí iframe elementu. Každé video má vedle sebe krátký popisek. U nějakých stránek může být ve spodní části kolotoč s logy partnerů jako na hlavní straně, popřípadě možnost prohlédnout si další videa vykreslená pomocí dlaždic.

Kontakt

Poslední podstránkou je kontaktní strana, na které se uživatel dozví několik základních informací o firmě a má možnost firmu kontaktovat pomocí e-mailu či mobilního telefonu. Stránka kontaktu obsahuje speciální komponentu s kontaktními informacemi. Na spodní části obrazovky je zafixován blok s kontaktním e-mailem a telefonem. Uživatele tedy tyto informace vidí po celou dobu prohlížení stránky.

Obrázek 10: Ukázka hlavní strany webu www.bervia.cz

4.3 Použité technologie

K účelům měření rychlosti byly vytvořeny dva testovací weby s doménami https://www.bervia-dev-old.eu a https://www.bervia-dev-new.eu. Na doméně „old“ je původní web a na doméně „new“ je spuštěn již optimalizovaný web.

Webové stránky jsou umístěny na webhostingu od firmy Wedos. Pro hosting byl použit balíček LowCost, což je dostačující volba pro naše využití.

Webové stránky používají technologie HTML, CSS a JavaScript. Jednotlivé stránky jsou statické a obsah se na nich tedy dynamicky nijak nemění. Web byl upravován v programu Visual Studio Code. Pro práci s CSS se používá preprocesor Sass, který je následně kompilován pomocí doplňku Sass přímo ve Visual Studiu Code.

4.4 Měření webu před optimalizacemi

Web byl měřen pomocí nástroje Lighthouse v DevTools v anonymním okně prohlížeče Google Chrome. Měření bylo provedeno zvlášť pro mobilní zařízení a pro desktop. Stránka se měřila vždy 10x z důvodu rozptylu jednotlivých měření. Pomocí nástroje Lighthouse bylo měřeno následujících 7 metrik: Performance, FCP, SI, LCP, TTI, TBT a CLS.

Hodnota performance vypočítává z jednotlivých metrik celkové hodnocení stránky. Maximální hodnota performance může být 100.

Měření na hlavní straně

	Performance	FCP	SI	LCP	TTI	TBT	CLS
Mobil	43,2	2,53	3,68	12,22	3,77	426	0,7282
Desktop	64,1	0,69	1,28	4,31	0,94	10	0,5459

Tabulka 3: Výsledky měření hlavní strany před optimalizacemi

Měření na straně portfolio

	Performance	FCP	SI	LCP	TTI	TBT	CLS
Mobil	57,3	2,58	3,7	2,82	6,73	562	0,789
Desktop	91,2	0,65	1,53	0,81	1,57	18	0,204

Tabulka 4: Výsledky měření strany portfolia před optimalizacemi

Měření na straně kontakt

Kontakt	Performance	FCP	SI	LCP	TTI	TBT	CLS
Mobil	72,7	2,48	2,52	2,55	3,39	261	0,7887
Desktop	94	0,57	0,65	0,57	0,7	0	0,2023

Tabulka 5: Výsledky měření strany kontakt před optimalizacemi

4.5 Optimalizace webu

Během analýzy webu bylo shledáno několik nedostatků a potencionálních možností ke zlepšení rychlosti načítání. Optimalizace s největším potenciálem zlepšení rychlosti a uživatelské přívětivosti při načítání byly aplikovány do zdrojového kódu stránky.

4.5.1 Minifikace

Nástroj Google Lightouse hlásil, že se na web načítají JavaScriptové soubory bez minifikace. Minifikace odstraňuje prázdné znaky z kódu a provádí další typy kompresí. Díky tomu se datová náročnost výsledného souboru značně zredukuje.

Kód byl minifikován pomocí online nástroje https://codebeautify.org/minify-html. Z důvodu těchto optimalizací se při načítání stránky ušetří stahování 75,7 kB.

Název souboru	Datová velikost před optimalizací	Datová velikost po optimalizaci	Rozdíl mezi velikostmi
Jquery.js	40,8 kB	31,2 kB	9,6 kB
Jquery.fancybox.js	39,4 kB	22,1 kB	17,3 kB
Owl.carousel.js	20,3 kB	11,6 kB	8,7 kB
Wow.js	3,7 kB	2,9 kB	0,8 kB
Lazysizes.js	6 kB	3,7 kB	2,3 KB
Bootstrap.bundle.js	48,8 kB	15,6 kB	33,2 kB
Typed.js	7,4 kB	3,6 kB	3,8 kB

Tabulka 6: Porovnávní velikosti js souborů před a po minifikaci

4.5.2 Přidání lazyloadingu

Webové stránky obsahují spoustu obrázků, které se stahují hned při úvodním načtení stránky.

Původně bylo zamýšleno pro lazyloading obrázků a iframů používat na stránkách html atribut loading. Výhodou využívání tohoto atributu je, že se nemusí používat žádná knihovna třetích stran a tudíž se ušetří stahování JavaScriptu knihovny. Nevýhodou je, že atribut loading nemá plnou podporu ve všech prohlížečích. Prohlížeče Safari a Firefox ho plně nepodporují.

```
<img src="/assets/images/about-us/graduation-cap.svg"
     loading="lazy" />
```

Obrázek 11: Vložení html atributu pro lazyloading

Následně bylo rozhodnuto, že se pro tzv. líné načítání využije externí knihovna třetí strany jquery.lazysizes.js. Knihovna má po minifikaci velikost 3,6 kB a na rychlost načtení by neměla mít velký vliv.

```
<img data-src="/assets/images/about-us/graduation-cap.svg"
     class="lazyload">
```

Obrázek 12: Využití externí knihovny pro lazyloading

Obrázkům a iframům byla přidána třída „lazyload“ a adresa obrázků byla vložena do atributu data-src místo src. Knihovna následně zajistí, aby se obrázky a iframy načetly až v případě, kdy jsou potřeba. Obrázky a iframy se stáhnou díky tomu, že je knihovna přesune z atributu data-src do src.

```
<img data-src="/assets/images/about-us/graduation-cap.svg"
     class="lazyloaded"
     src="/assets/images/about-us/graduation-cap.svg">
```

Obrázek 13: Ukázka img elementu po načtení obrázku pomocí lazyloadingu

Tímto krokem se na každé stránce stahuje výrazně menší množství dat při úvodním načtení. Největší posun byl zaznamenán na stránce portfolia. Bez líného načítání stránka musí stáhnout 18,1 MB dat. Po použití líného načítání stránka stáhne pouze 7,5 MB a zbylá část se stáhne až v případě, kdy dojde k posunu po stránce.

4.5.3 Optimalizace velikosti a datové náročnosti obrázků

Na stránkách se používaly velké a neoptimalizované obrázky. Obrázky byly zmenšeny na maximální rozměry, ve kterých se vykreslují na webových stránkách. Následně byly datově optimalizovány pomocí online služby https://compresspng.com/.

Například obrázek na hlavní straně v hlavním banneru měl původně 2,2 MB a stažení trvalo na rychlém 5G připojení 520 ms. Po zmenšení rozměrů a datové optimalizaci je datová náročnost obrázku 294 kB a stažení trvalo pouze 62 ms.

Name	Type ▲	Size	Time
main-banner-image-dark-desktop.jpg	jpeg	2.2 MB	520 ms

Obrázek 14: Datová náročnost obrázku hlavního banneru před optimalizací

Datová náročnost obrázku byla tedy zmenšena o téměř 87 %. Podobné míry komprimace jsou i u všech ostatních obrázků.

Name	Type	Size	Time
main-banner-image-dark.jpg	jpeg	296 kB	62 ms

Obrázek 15: Datová náročnost obrázku hlavního banneru po optimalizaci

4.5.4 Úpravy načítání JavaScriptů na stránku

Načítání skriptů na web bylo původně umístěno hned pod začátkem tagu <body>. Kvůli této úpravě se čekalo na stažení skriptů a až následně se mohlo zparsovat HTML a zobrazit na stránce.

```
▼<body class>
    <script src="/assets/js/vendor/jquery.js"></script>
    <script src="/assets/js/vendor/jquery.fancybox.js"></script>
    <script src="/assets/js/vendor/owl.carousel.js"></script>
    <script src="/assets/js/vendor/wow.js"></script>
    <script src="/assets/js/vendor/lazysizes.js"></script>
    <script src="/assets/js/vendor/bootstrap/bootstrap.bundle.js"></script>
    <script src="/assets/js/vendor/typed.js"></script>
  ▶<header class="header wow fadeIn header-hp">…</header>
  ▶<main>…</main>
  ▶<footer class="section section--padding section--footer wow fadeInUp">…</footer>
  ▶<script>…</script>
  </body>
```

Obrázek 16: Html struktura webu před optimalizací načtení skriptů

Skripty byly přesunuty až na konec tagu <body>. Díky této úpravě se nyní první zparsuje HTML a až následně se budou stahovat skripty.

```
▼<body class>
  ▶<header class="header wow fadeIn header-hp">…</header>
  ▶<main>…</main>
  ▶<footer class="section section--padding section--footer wow fadeInUp">…</footer>
    <script src="/assets/js/vendor/jquery.js"></script>
    <script src="/assets/js/vendor/lazysizes.min.js"></script>
    <script src="/assets/js/vendor/owl.carousel.min.js"></script>
    <script src="/assets/js/vendor/wow.min.js"></script>
    <script src="/assets/js/vendor/lazysizes.min.js"></script>
    <script src="/assets/js/Shared/all.js"></script>
    <script src="/assets/js/vendor/bootstrap/bootstrap.min.js"></script>
    <script src="/assets/js/vendor/typed.min.js"></script>
  ▶<script>…</script>
    <script> new WOW().init(); </script>
  </body>
```

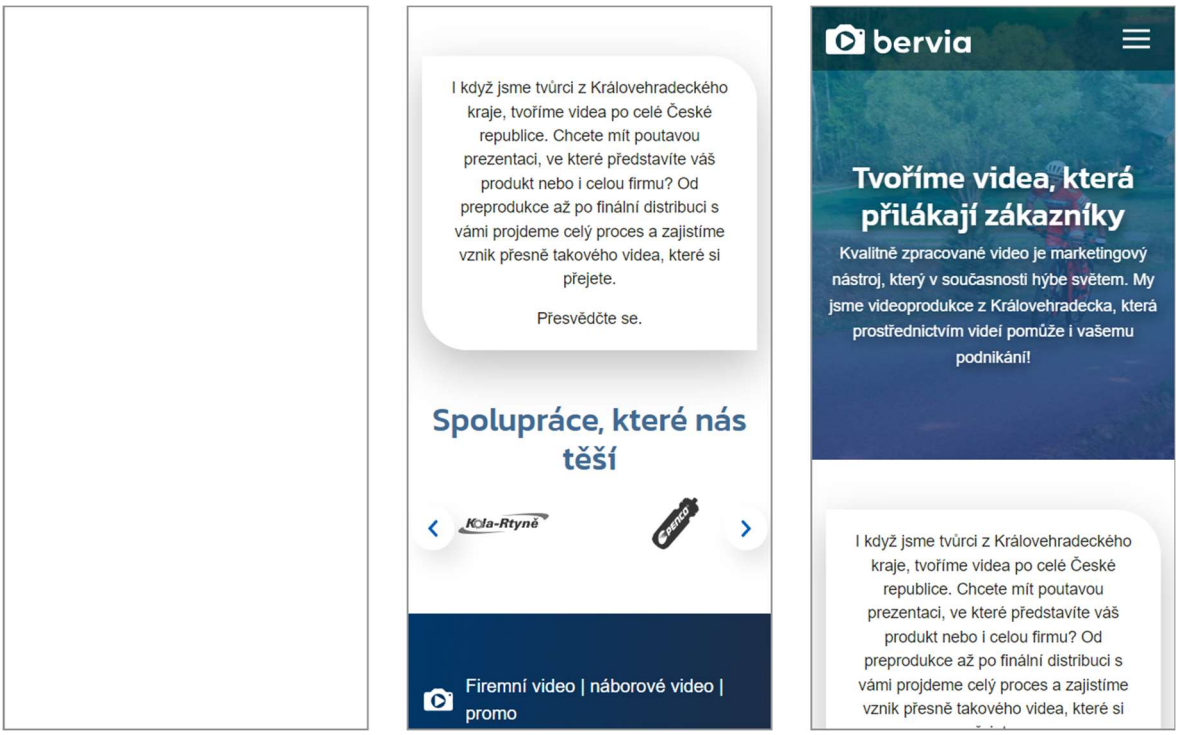
Obrázek 17: Html struktura webu po optimalizací načtení skriptů

Zároveň část scriptů pro nastavení owlCarouselu a nastavení funkce pro scroll byla původně umístěna přímo v tagu <script> uvnitř HTML dokumentu. Veškerý kód, který nemusel nezbytně nutně zůstat přímo v HTML dokumentu byl přesunut do souboru all.js a soubor byl zminifikován.

4.5.5 Vykreslení hlavního banneru

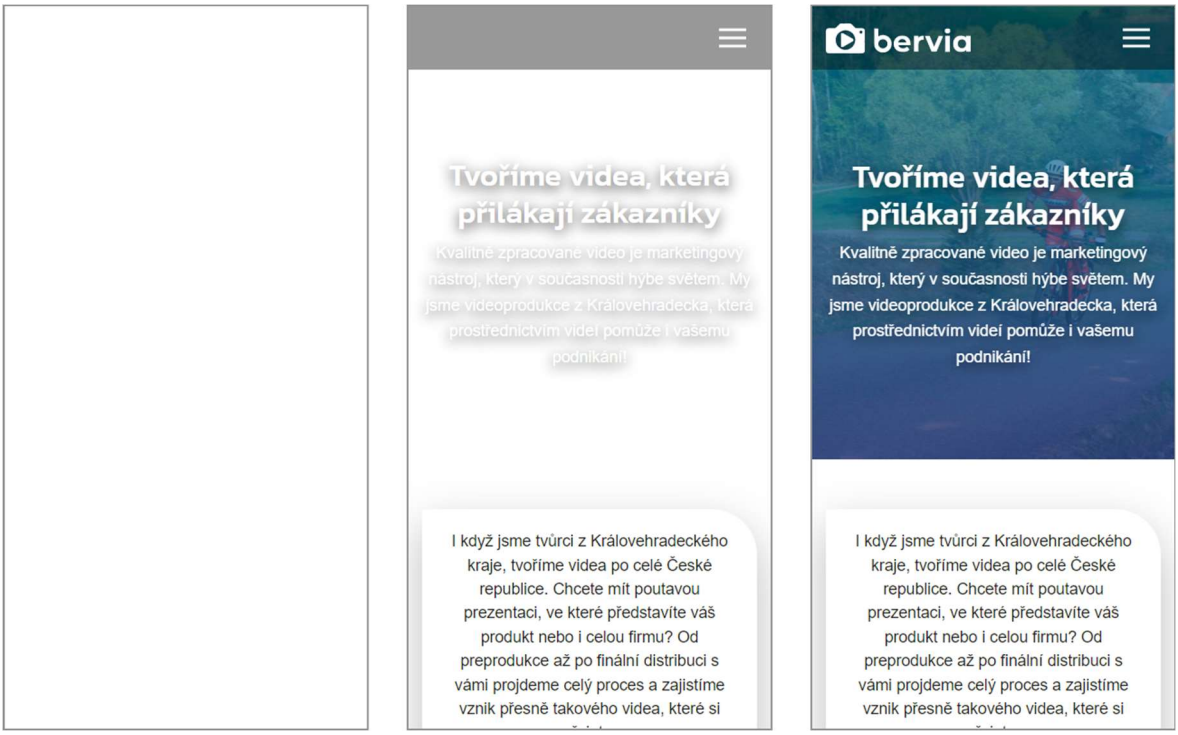
Banner na hlavní straně byl původně inicializován pomocí JavaScriptové knihovny owlCarousel. Nevýhodou využití této knihovny pro vykreslení banneru je, že se HTML kód vygeneruje až po inicializaci a zpracování JavaScriptu.

Kvůli této prodlevě vzniká posun layoutu na stránce, což signalizují zvýšené hodnoty metriky CLS. Při načtení se nejdříve banner vykreslí s nulovou výškou a až po zpracování JavaScriptu se zobrazí ve správných rozměrech.



Obrázek 18: Ukázka postupného vykreslení hlavního banneru před optimalizací

Díky tomu, že se v banneru používá pouze jeden obrázek, tak nedává smysl pro vykreslení používat owlCarousel. Hlavní banner byl překódován tak, aby se vykresloval pouze pomocí HTML a CSS. Po optimalizaci se banner zobrazí již se správnými rozměry a postupně se pouze donačítají data jako obrázky, font atp.



Obrázek 19: Ukázka postupného vykreslení hlavního banneru po optimalizací

4.5.6 Hlavní banner měl původně tři obrázky, nyní jeden

Banner na hlavní straně původně obsahoval 3 obrázky. Pro desktop, tablet a mobilní zařízení. Mezi obrázky byl rozdíl pouze v rozměrech. Důvodem bylo, aby byl výřez obrázku vždy umístěn na postavu cyklisty.

```
<div class="main-banner">
  <img class="main-banner__image main-banner__image--desktop" src="/assets/images/main-banner-image-dark-desktop.jpg">
  <img class="main-banner__image main-banner__image--tablet" src="/assets/images/main-banner-image-dark-tablet.jpg">
  <img class="main-banner__image main-banner__image--mobile" src="/assets/images/main-banner-image-dark-mobile.jpg">
  <div class="main-banner__item-overlay"></div>
  <div class="main-banner__item-inner">...</div>
</div>
```

Obrázek 20: Html struktura obrázku hlavního banneru před optimalizací

Banner byl optimalizován tak, aby stránka nemusela stahovat 3 obrázky, ale nově tam je pouze jeden obrázek, který mění poměr stran při responzivitě pomocí CSS. Obrázek je zároveň pozicován pomocí absolutní pozice tak, aby postava cyklisty byla vždy ve výřezu obrázku vykreslován na stránce.

```
<div class="main-banner">
  <img class="main-banner__image" src="/assets/images/main-banner-image-dark.jpg">
  <div class="main-banner__item-overlay"></div>
  <div class="main-banner__item-inner">...</div>
</div>
```

Obrázek 21: Html struktura obrázku hlavního banneru po optimalizací

4.5.7 Zrušení owlCarouselu na podstránkách

Na podstránkách webu byl banner tvořen pomocí owlCarouselu podobně jako banner na hlavní straně. Banner na stránce taktéž vykazoval posun layoutu kvůli inicializaci pomocí JavaScriptu.

Podstránky obsahovaly pouze jednu položku v banneru, tudíž mohly být aplikovány podobné úpravy jako u banneru na hlavní straně. Banner byl překódován a nově se vykresluje pouze pomocí HTML a CSS.

Po provedených optimalizacích banner při načtení stránek neposkakuje a díky tomu dochází zejména ke zlepšení metriky CLS.

4.5.8 Zrušení stahování zbytečných fontů

Původní web používal pro nadpisy 5 řezů fontu „Kanit“. Určité řezy fontu byly využity na stránkách ojediněle a nebylo optimální, aby každá stránka stahovala řez fontu, který se využívá například pouze na jednom místě na webu.

Name	Type	Size	Time
Kanit-Medium.woff	font	73.5 kB	179 ms
Kanit-Regular.woff	font	73.0 kB	212 ms
Kanit-Light.woff	font	72.1 kB	198 ms
Kanit-Bold.woff	font	73.7 kB	179 ms
Kanit-ExtraBold.woff	font	74.6 kB	213 ms

Obrázek 22: Ukázka načítaných řezů fontu před optimalizací

Nadpisy byly sjednoceny na řez písma medium. Tento řez byl následně i optimalizován pomocí služby https://www.fontsquirrel.com/.

Name	Type	Size	Time
kanit-medium-webfont.woff	font	18.5 kB	116 ms

Obrázek 23: Ukázka načítaných řezů fontu po optimalizaci

Web původně stahoval 366,9 kB fontů. Po provedených optimalizacích stahuje 18,4 kB a bylo tedy ušetřeno 348 kB při načítání stránky.

5 Výsledky a diskuse

Veškerá měření proběhla před zapracováním optimalizací a následně až po aplikování všech optimalizací. Z tohoto důvodu nelze vyhodnotit, jaká optimalizace měla jaký vliv, na jakou stránku a na jakou metriku.

Vyhodnocovat se bude tedy pouze celkový vliv všech aplikovaných optimalizací na jednotlivé stránky pro desktopovou verzi a mobilní verzi.

Hlavní strana

	Performance	FCP	SI	LCP	TTI	TBT	CLS
Mobil - před	43,2	2,53	3,68	12,22	3,77	426	0,7282
Mobil - po	96,1	1,63	1,97	1,65	2,6	170	0
Desktop - před	64,1	0,69	1,28	4,31	0,94	10	0,5459
Desktop - po	99,8	0,4	0,65	0,48	0,54	8	0,039

Tabulka 7: Porovnání výsledků měření na hlavní straně

Strana portfolio

	Performance	FCP	SI	LCP	TTI	TBT	CLS
Mobil - před	57,3	2,58	3,7	2,82	6,73	562	0,789
Mobil - po	73,7	1,68	2,94	1,85	3,6	523	0
Desktop - před	91,2	0,65	1,53	0,81	1,57	18	0,204
Desktop - po	98,4	0,4	1	0,5	0,58	10	0,0085

Tabulka 8: Porovnání výsledků měření na straně portfolio

Strana kontakt

	Performance	FCP	SI	LCP	TTI	TBT	CLS
Mobil - před	72,7	2,48	2,52	2,55	3,39	261	0,7887
Mobil - po	98,8	1,64	1,65	1,69	2,36	50	0
Desktop - před	94	0,57	0,65	0,57	0,7	0	0,2023
Desktop - po	100	0,4	0,52	0,41	0,48	0	0,0052

Tabulka 9: Porovnání výsledků měření na straně kontakt

Optimalizace byly z velké části úspěšné, jelikož na všech měřených stranách došlo k výraznému zlepšení všech měřených metrik.

Metrika FCP vykazuje, že u všech třech měřených stran by první smysluplný obsah na mobilu měl být podle laboratorních dat vykreslen až o jednu sekundu dříve.

Další velké zlepšení pro uživatele signalizuje metrika LCP. Díky ní by se měl největší obsah na stránce vykreslit mnohem rychleji než na starém webu. Největší zlepšení je vidět na hlavní straně, kde původně načtení největšího obsahu trvalo 12,22 s a po zapracování optimalizací trvá načtení pouze 1,65 s.

Téměř na všech stranách se podařilo snížit metriku CLS k hodnotě nula. Díky tomu budou uživatelé zažívat mnohem příjemnější uživatelský zážitek, jelikož jim stránka nebude při načítání poskakovat, ale každý prvek si bude udržovat svojí pozici i rozměry.

Metrika performance vypočítává svoji hodnotu ze všech ostatních metrik. Maximální možná hodnota této metriky je 100. V případě, že výsledek měření má hodnotu blízkou 100 bodů, tak to značí, že je daná strana z pohledu rychlosti načítání dostatečně optimalizovaná a již není nutné dělat další optimalizace.

U všech stran kromě mobilní verze strany portfolia se podařilo metriku přiblížit ke 100 bodům, což signalizuje, že rychlosti načtení jednotlivých stran jsou velmi dobré.

Mobilní verze strany portfolia má hodnotu 73,7 bodů. V porovnání s konkurencí se jedná stále o nadprůměrný výsledek, ale bylo by možné provést další kroky ke zlepšení. Mezi další optimalizace by se mohlo řadit například přesunutí ukázkových videí do interního úložiště místo načítání pomocí iframu. Pro tuto stranu se doporučuje provést detailnější analýzu, aby bylo možné navrhnout i další optimalizace, které by vedly ke zlepšení rychlosti načtení a uživatelského zážitku z načítání´

6 Závěr

V teoretické části bylo popsáno z čeho se skládají webové stránky. Následně byly analyzovány nejpoužívanější frameworky pro tvorbu webových stránek a jejich negativní dopad na rychlost načítání. Většina teoretické části se zabývá analýzou metrik rychlosti načtení, nástroji pro měření daných metrik a tipy pro zrychlení webových stránek.

K prokázání poznatků z teoretické části byla v praktické části analyzována již existující webová stránka www.bervia.cz, která se zabývá tvorbou videí. Z analýzy vzešly tipy na optimalizace webové stránky. Ty, které měly největší potenciál ke zlepšení daných metrik byly aplikovány. Mezi hlavní optimalizace patří: minifikace souborů, přidání lazyloadingu pro obrázky a iframy, optimalizace velikosti a kvality obrázků, změna vykreslení hlavního banneru z důvodu poskakování při načítání, zrušení stahování zbytečných řezů fontu a datová optimalizace fontů.

K měření rychlosti byl použit nástroj Google Lighthouse v anonymním okně prohlížeče Google Chrome v devtools. Byly měřeny 3 strany webu: hlavní strana, strana portfolio a strana kontakt. Každá strana byla měřena na mobilní verzi i na desktopové verzi. Z důvodu drobných odchylek měření byla každá strana změřena 10krát a následně byly výsledky aritmeticky zprůměrovány.

V poslední části byly porovnány výsledky měření před a po optimalizacích. U všech stran kromě mobilní verze strany portfolia došlo k takovému zlepšení, že bylo konstatováno, že již není potřeba dělat žádné další optimalizace. U mobilní verze strany portfolia byla doporučena podrobnější analýza rychlosti načítání této strany.

Jelikož se metriky načtení i dané webové stránky stále vyvíjí, tak se doporučuje zaregistrovat webové stránky do analytického nástroje, který bude měřit rychlost v pravidelných intervalech a majitele webových stránek bude pravidelně informovat o stavu rychlosti načítání a uživatelské přívětivosti při načítání.

8 Seznam použitých zdrojů

(1) HTML5. In: *MDN Web Docs* [online]. MDN contributors, c1998–2022 [cit. 2022-08-14]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/HTML5

(2) MAURICE, Florence. *HTML und CSS Für Dummies* [online]. John Wiley & Sons, Incorporated, 2019 [cit. 2022-12-13]. ISBN 9783527821259. Dostupné z: https://ebookcentral-proquest-com.infozdroje.czu.cz/lib/czup/detail.action?docID=5906779&query=9783527821259

(3) CSS reference. In: *MDN Web Docs* [online]. MDN contributors, c1998–2022 [cit. 2022-08-14]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

(4) LARSEN, Rob a Rob LARSEN. *Beginning HTML and CSS* [online]. 1. John Wiley & Sons, Incorporated, 2012 [cit. 2022-12-13]. ISBN 9781118340288. Dostupné z: https://ebookcentral-proquest-com.infozdroje.czu.cz/lib/czup/detail.action?docID=1138437&query=9781118340288

(5) ŘEZÁČ, Jan. *Web ostrý jako břitva: Návrh fungujícího webu pro webdesignery a zadavatele projektů.* [online]. Baroque partners, 2014 [cit. 2022-12-13]. ISBN 9788087923016.

(6) ČÁPKA, David. Lekce 1 - Úvod do JavaScriptu. In: *IT Network* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk

(7) PERNA, Maria Antonietta. 3 Tips for Speeding Up Your Bootstrap Website. In: *Sitepoint* [online]. c2000-2022 [cit. 2022-12-13]. Dostupné z: https://www.sitepoint.com/3-tips-to-speed-up-your-bootstrap-website/

(8) SVOBODA, Radek. Nejpoužívanější softwarové frameworky pro vývoj webových aplikací. In: *Cool club: Club pro IT odborníky* [online]. 2020 [cit. 2022-08-14]. Dostupné z: https://club.coolpeople.cz/nejpouzivanejsi-softwarove-frameworky-pro-vyvoj-webovych-aplikaci/1379.html

(9) ČÁPKA, David. Lekce 1 - Úvod do jQuery. In: *IT Network* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.itnetwork.cz/javascript/jquery-zaklady/javascript-tutorial-funkcionalni-programovani-a-jquery-webova-kalkulacka

(10) WALTON, Philip. Web Vitals. In: *Web.dev* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://web.dev/vitals/

(11) UZAYR, Sufyan bin. *Web Performance Optimization: A Practical Approach* [online]. Taylor & Francis Group, 2022 [cit. 2022-12-13]. ISBN 9781000541342. Dostupné z: https://ebookcentral-proquest-com.infozdroje.czu.cz/lib/czup/detail.action?docID=6868919&query=9781000541342

(12) WALTON, Philip. Largest Contentful Paint (LCP). In: *Web.dev* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://web.dev/lcp/

(13) WALTON, Philip. First Input Delay (FID). In: *Web.dev* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://web.dev/fid/

(14) MICHÁLEK, Martin. Metrika „Kumulativní posun layoutu“ (Cumulative Layout Shift, CLS). In: *Vzhůru dolů* [online]. [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/metrika-cls

(15) WALTON, Philip a Milica MIHAJLIJA. Cumulative Layout Shift (CLS). In: *Web.dev* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://web.dev/cls/

(16) SHIVAKUMAR, Shailesh Kumar. *Modern Web Performance Optimization: Methods, Tools, and Patterns to Speed up Digital Platforms* [online]. Apress L. P., 2020 [cit. 2022-12-13]. ISBN 9781484265284. Dostupné z: https://ebookcentral-proquest-com.infozdroje.czu.cz/lib/czup/detail.action?docID=6408038&query=9781484265284

(17) WAGNER, Jeremy a Barry POLLARD. Time to First Byte (TTFB). In: *Web.dev* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://web.dev/ttfb/

(18) MICHÁLEK, Martin. Co je „Doba do načtení prvního bajtu“ (aneb „Time To First Byte“ aneb TTFB)?. In: *Vzhůru dolů* [online]. [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/ttfb

(19) WALTON, Philip. First Contentful Paint (FCP). In: *Web.dev* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://web.dev/fcp/

(20) WALTON, Philip. Total Blocking Time (TBT). In: *Web.dev* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://web.dev/tbt/

(21) WALTON, Philip. Time to Interactive (TTI). In: *Web.dev* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://web.dev/tti/

(22) About PageSpeed Insights. In: *PageSpeed Insights* [online]. 2022 [cit. 2022-08-14].

Dostupné z: https://developers.google.com/speed/docs/insights/v5/about

(23) Metric Distribution. In: *Google Developers* [online]. [cit. 2022-08-15]. Dostupné z: https://developers.google.com/speed/docs/insights/v5/about

(24) MICHÁLEK, Martin. Chrome UX Report: Data o rychlosti webu přímo od uživatelů. In: *Vzhůru dolů* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/chrome-ux-report

(25) MICHÁLEK, Martin. Lighthouse: audit webu od Google. In: *Vzhůru dolů* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/lighthouse

(26) HTTP. In: *MDN Web Docs* [online]. MDN contributors, c1998–2022 [cit. 2022-08-14]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP

(27) An overview of HTTP. In: *MDN Web Docs* [online]. MDN contributors, c1998–2022 [cit. 2022-08-14]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

(28) MICHÁLEK, Martin. Rychlý protokol HTTP/2: S nasazením na weby na nic nečekejte. In: *Vzhůru dolů* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/http-2

(29) MICHÁLEK, Martin. HTTP/3: Slibná nová verze protokolu, zatím ale jen s malou podporou. In: *Vzhůru dolů* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/http-3

(30) MICHÁLEK, Martin. Lazy loading v kontextu webového frontendu: co to je a proč to dělat?. In: *Vzhůru dolů* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/lazy-loading

(31) MICHÁLEK, Martin. Preload: Přednačtení prvků stránky detailně a do hloubky. In: *Vzhůru dolů* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/preload

(32) Resource Hints: preload. In: *Can I Use* [online]. [cit. 2022-08-14]. Dostupné z: https://caniuse.com/?search=preload

(33) Preload support. In: *Can I Use* [online]. [cit. 2022-08-15]. Dostupné z: https://caniuse.com/?search=preload

(34) MICHÁLEK, Martin. CSS: Optimalizace datové velikosti. In: *Vzhůru dolů* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-

optimalizace

(35) MICHÁLEK, Martin. Critical CSS a zrychlení zobrazení stránky. In: *Vzhůru dolů* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/blog/35-critical-css

(36) MICHÁLEK, Martin. Vkládání JavaScriptu jako async, defer a type="module" versus rychlost webu. In: *Vzhůru dolů* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/js-async-defer-module

(37) MICHÁLEK, Martin. Průvodce formáty obrázků pro web: JPEG, WebP, AVIF, PNG, GIF a SVG. In: *Vzhůru dolů* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/obrazky-formaty

(38) MICHÁLEK, Martin. Deskriptor font-display: Jak na řízené vykreslování webfontů?. In: *Vzhůru dolů* [online]. 2022 [cit. 2022-08-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-font-display

9 Seznam obrázků, tabulek a zkratek

9.1 Seznam obrázků

9.2 Seznam tabulek

9.3 Seznam použitých zkratek

CSS – Cascading Styl Sheets

JS – JavaScript

DOM – Document Object Model

HTML – Hypertext Markup Language

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

PNG – Portable Network Graphics

JPG – Joint Photographic Experts Group