VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná Informatika

VYTVOŘENÍ WEBOVÝCH STRÁNEK PRO KOSMETICKÝ SALÓN

Bakalářská práce

Autor práce: Robin Hubáček

Vedoucí práce: Mgr. Hana Vojáčková, Ph.D.

Jihlava 2024

Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce:	**Robin Hubáček**
Studijní program:	Aplikovaná informatika
Obor:	Aplikovaná informatika
Garant studijního programu:	Ing. Lenka Kuklišová Pavelková, Ph.D.
Název práce:	**Vytvoření webových stránek pro kosmetický salón**
Vedoucí práce:	Mgr. Hana Vojáčková, Ph.D.
Cíl práce:	Práce se zaměří na vytvoření webových stránek s rezervačním systémem pro osobu podnikající v oblasti kosmetiky a vlasové péče. Cílem práce je analyzovat současný stav webu, navrhnout a implementovat nový design a funkcionality, které by měly vést ke zvýšení návštěvnosti a zlepšení uživatelského zážitku. Budou zde rozebrány základní principy tvorby webových stránek. Práce bude obsahovat proces návrhu a vývoje nového webu, testování a vyhodnocení jeho úspěšnosti. Výsledkem bude funkční a moderní webová stránka, která bude sloužit jako efektivní nástroj pro propagaci a prodej služeb v oblasti kosmetiky a vlasové péče.

Abstrakt

Bakalářská práce se zaměřuje na vytvoření webových stránek pro kosmetický salón s cílem modernizovat jeho online přítomnost a efektivně propagovat kosmetické služby. Zaměřuji se na využití moderních technologií a vytvoření rezervačního systému. Práce zahrnuje analýzu stávajícího webu, návrh nového designu, vytvoření a implementace nového webu, testování a vyhodnocení úspěšnosti projektu. Cílem je vytvořit moderní, funkční a uživatelsky přívětivou webovou stránku podporující rozvoj kosmetického salónu v online prostředí. Výsledkem práce je funkční webová stránka s rezervačním systémem.

Klíčová slova

HTML; CSS; SCSS; JavaScript; JQuery; PHP; SQL; Rezervační systém; npm; Webpack;

Abstract

The bachelor thesis focuses on creating a website for a beauty salon to modernize its online presence and effectively promote its beauty services. i focus on the use of modern technology and the creation of a booking system. The work includes analysis of the existing website, designing a modern design, creating, and implementing a new website, testing, and evaluating the success of the project. The goal is to create a modern, functional, and user-friendly website, supporting the development of the beauty salon in the online environment. The result of the work is a functional website with a booking system.

Keywords

HTML; CSS; SCSS; JavaScript; PHP; SQL; Booking system; npm; Webpack

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 3. dubna 2024

…….

Podpis studenta/ky

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářské práce, své rodině za podporu při studiu, a také své známé, která mi poskytla odbornou korekturu.

Obsah

Seznam obrázků

Seznam zkratek

API – Application Programming Interface

CSS – Cascading Style Sheets

HTML – Hyper Text Markup Language

npm – Node Package Manager

PHP – Hypertext Preprocessor

SQL – Structured Query Language

Úvod

Bakalářská práce se hlouběji věnuje problematice tvorby webových stránek pro kosmetický salón, což je důležité téma reagující na aktuální trendy a potřeby v digitální době. Mým základním cílem je vytvořit efektivní online nástroj, který bude sloužit jako prostředek propagace a prodeje kosmetických služeb pro podnikatele v oblasti kosmetiky a vlasů.

Motivace pro volbu problematiky spočívá v pozorování současných trendů v oboru, kde online přítomnost a dostupnost informací hrají klíčovou roli v konkurenčním prostředí. Rovněž je zde potřeba reagovat na konkrétní potřeby podnikatele, který si uvědomuje význam modernizace svého online zastoupení. Moje studijní zaměření mi navíc umožňuje spojit teoretické poznatky z oblasti tvorby webových stránek s praktickým uplatněním, což zvyšuje relevanci práce.

Význam práce spočívá nejen v poskytnutí konkrétního řešení pro konkrétního podnikatele, ale také v přínosu pro obecné poznání o tvorbě moderních webových stránek a využití digitálních nástrojů pro podporu podnikání. Pro dosažení mých cílů budu využívat moderní technologie, včetně HTML, CSS a JavaScript, a pracovat s databázovým systémem pro správu informací o službách a rezervacích. K návrhu stránek použiji grafický nástroj Figma.

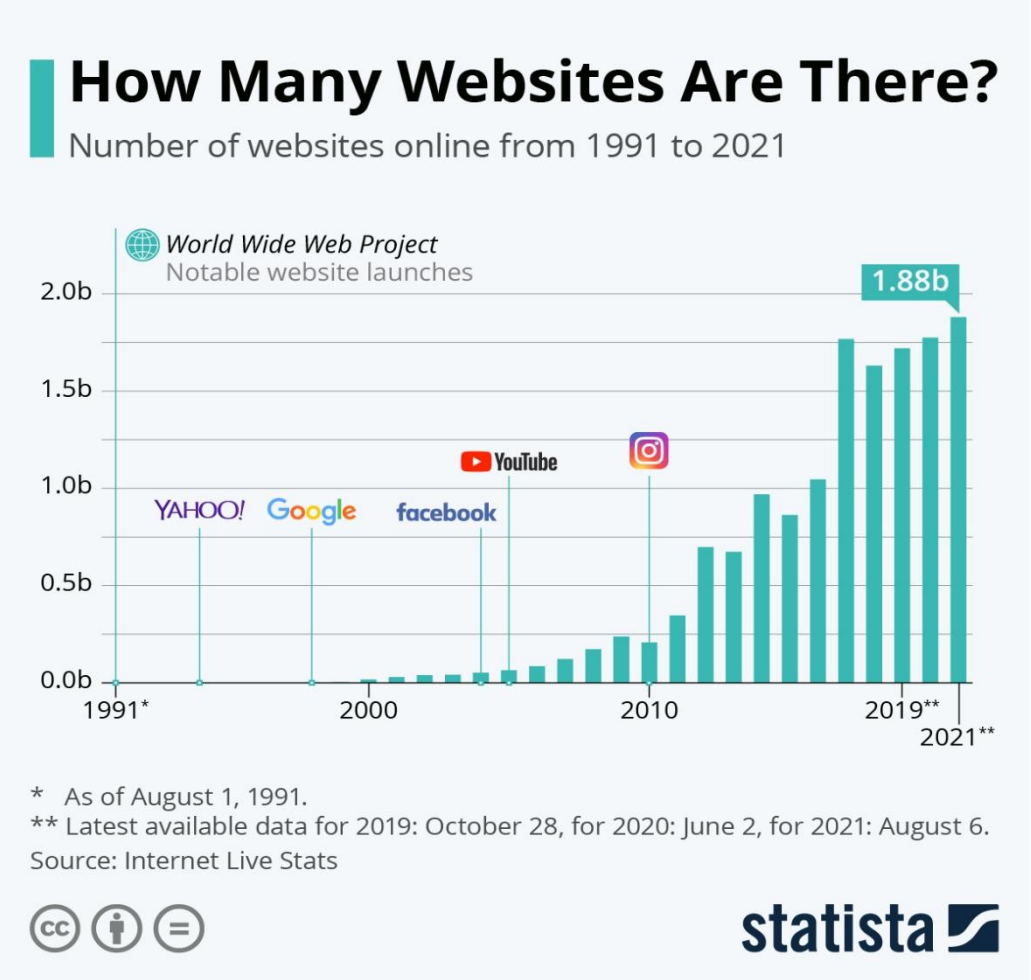
V následujících kapitolách se budu zabývat analýzou stávajícího webu, návrhem a implementací nového designu a rezervačního systému. Samozřejmě, nedílnou součástí práce bude i testování a vyhodnocení úspěšnosti mého projektu. Cílem je vytvořit moderní, funkční a uživatelsky přívětivou webovou stránku, která efektivně slouží pro podporu a rozvoj kosmetického salonu v online prostředí a přispívá k jeho konkurenceschopnosti na trhu.

Výstupem práce bude funkční webová stránka s rezervačním systémem, která bude kódována v hypertextovém značkovacím jazyce HTML. Jeho designová úprava bude řešena pomocí kaskádových stylů, tedy CSS, a pro dynamiku webu budu využívat programovací jazyk JavaScript a programovací jazyk PHP. Data o rezervacích budu ukládat do databázových tabulek. Pro usnadnění práce a zlepšení organizace v tomto projektu aplikuji Webpack a pro jeho instalaci využiji správce balíčků npm.

1 Moderní vývoj webu

V dnešní době je vývoj webových stránek velmi žádaný, ačkoliv ho rozvoj nových technologií dělá těžším a těžším. Vývoj webových stránek spočívá v tvoření stránek na internet za pomoci různých nástrojů, od jednoduchých po komplexnější webové stránky.

Následující graf ukazuje počet webových stránek, které existují a vyskytují se na internetu k roku 2021. Obsahuje také, v jakých letech byly vytvořeny jedny z nejpopulárnějších, jako je Youtube, Facebook a podobně.



Obr. 1: Množství webových stránek

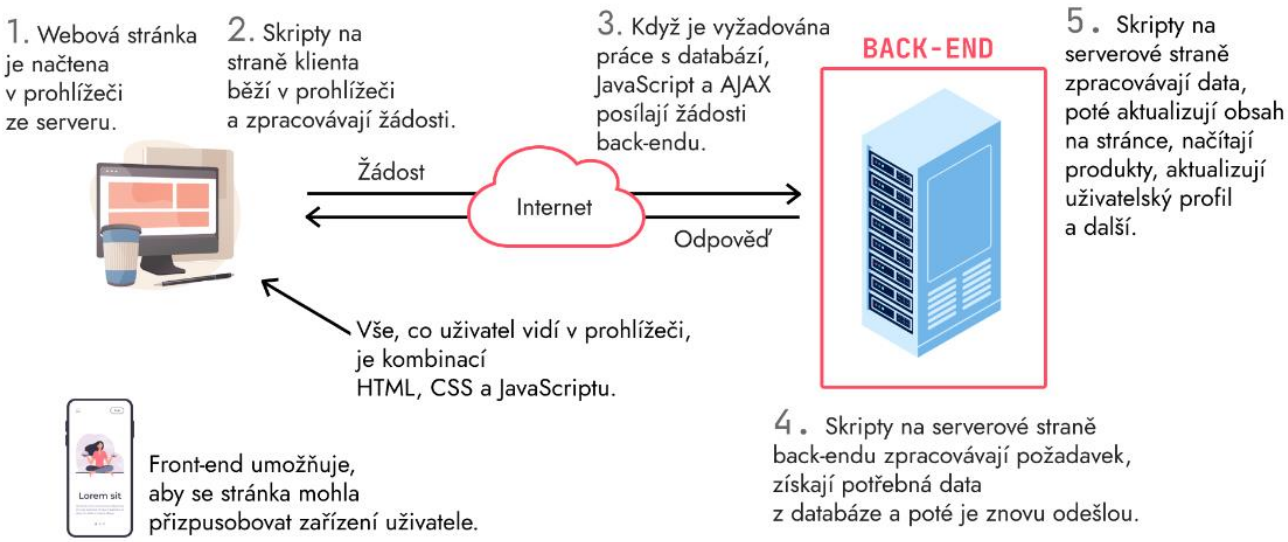
Zdroj: (Armstrong 2011)

Charakteristiky webového vývoje:

- Jasná a uživatelsky přívětivá navigace
- Nabízí aktuální obsah
- Plný nových nápadů
- Jedinečný oproti ostatním
- Zabezpečený
- Přístupné informace

1.1 Front-end development

Front-end, často známý jako “klientská část”, představuje vizuální obsah, neboli to, co vidí uživatel, když se dostane na naši webovou stránku. Na naší webové stránce uživatel může interagovat s různými prvky, příkladem může být rezervace do kosmetického salónu, kde si klient (uživatel) vybere službu, přidá ji do rezervace a poté rezervaci odešle.



Obr. 2: Architektura client-server

Zdroj: Vlastní

Technologie, které patří do tohoto odvětví front-endu, jsou HTML (HyperText Markup Language), CSS (Cascading Style Sheets) a programovací jazyk JavaScript, který nám umožňuje dodat webové stránce interakci, jinak řečeno dynamiku. Tyto vyjmenované technologie jsou jedny z nejzákladnějších, které by měl front-end vývojář ovládat, pokud se jím chce stát. Existuje zde mnoho dalších technologií, které jsou pokročilejší a jejich využití závisí čistě na individuálních potřebách. Mezi tyto nástroje se řadí frameworky a preprocesory. Framework je balíček různých vývojářských nástrojů, které nám pomáhají ve vývoji aplikace pomocí předdefinovaných funkcí. Nejpopulárnější frameworky jsou: React, Angular, Vue.js a mnoho dalších. Preprocesor je nástroj nebo jazyk, který umožňuje vývojářům psát kód s některými výhodami a abstrakcemi, jež nejsou k dispozici v cílovém jazyce. Tyto nástroje potom převedou (předzpracují) kód napsaný ve zvláštní syntaxi nebo s rozšířenými funkcemi do finální podoby, kterou lze použít v cílovém prostředí. *(Lindley, 2023)*

Příklady preprocesorů:

- SCSS
- Less
- TypeScript
- CoffeScript

Abychom mohli vidět obsah stránek a společné fungování všech těchto technologií společně, musíme mít webový prohlížeč. Webový prohlížeč je software, který získává, zpracovává, prezentuje a odesílá informace.

Webové prohlížeče jsou:

- Internet Explorer
- Chrome
- Firefox
- Safari

1.2 Back-end development

Back-end, jinak řečeno serverová část, představuje část webové aplikace nebo softwarového systému, běží na serveru a zajišťuje logiku, interakci s databází a všechny operace, které probíhají na straně serveru. Back-end je zodpovědný za zpracování dat a jejich odeslání na front-endovou (klientskou) část.

Back-end může zajišťovat:

- Databázové operace
- Autentizaci a autorizaci
- API (rozhraní pro programování aplikací)
- Bezpečnou komunikaci mezi klientem a serverem
- Správu chyb

Back-end vývojář pracuje s různými programovacími jazyky, které mohou být: Python, PHP, Java, Node.js a další. Stejně jako front-end, i back-end má své frameworky. Dovednost, kterou vývojář v této oblasti musí ovládat, je práce s databázemi. Databáze nám pomáhají uchovávat data, k nimž poté můžeme přistupovat skrze určité rozhraní a dále s nimi pracovat. *(Coursera Inc., 2023)*

Mezi databázové nástroje patří:

- MySQL
- PostreSQL
- Microsoft SQL Server
- MongoDB
- Cassandra
- Firebase

1.3 Full-stack development

Fullstack development označuje schopnost vývojáře pracovat na obou stranách (front-end a back-end) webové aplikace nebo softwarového produktu. Fullstack developer má tedy dovednosti a znalosti potřebné pro práci jak s uživatelským rozhraním (front-end), tak se serverovou částí a databázemi (back-end).

Fullstack developer může pracovat na kompletním vývoji projektu od začátku až do konce, nebo se specializovat na určitou oblast (front-end nebo back-end) a spolupracovat s ostatními specialisty. Tato široká paleta dovedností umožňuje fullstack developerovi efektivněji komunikovat mezi oběma stranami vývoje a lépe porozumět celkovému kontextu projektu. *(Simplilearn Solutions, 2023)*

2 Technologie

V průběhu vývoje mé webové stránky byla systematicky využívána široká paleta moderních technologií, které umožnily dosáhnout zamýšlených cílů a vytvořit komplexní a inovativní řešení. Následující kapitoly zahrnují klíčové technologie a jejich funkce, které byly implementovány a integrovány do vlastního projektu.

2.1 Figma

Figma je moderní nástroj pro design a prototypování, který se stal velmi populárním mezi designéry a týmy pracujícími na vývoji digitálních produktů. Jeho významná vlastnost spočívá v tom, že je založený na cloudovém přístupu, což umožňuje spolupráci v reálném čase mezi členy týmu bez ohledu na jejich fyzickou polohu.

Jedním z hlavních prvků, kterým se Figma odlišuje, je jeho schopnost umožňovat týmovou spolupráci na jediném designovém projektu. Uživatelé mohou synchronizovat své úpravy a vidět je okamžitě, což výrazně zjednodušuje proces sdílení nápadů a zpřesňování designu.

Figma nabízí robustní sadu nástrojů pro vytváření rozsáhlých wireframů, designů uživatelských rozhraní a interaktivních prototypů. Tato funkčnost je klíčová pro iterativní proces designu, kde mohou členové týmu společně pracovat na vylepšení a ladění návrhů.

Další výhodou Figma je jeho přístupnost. Uživatelé nemusí instalovat žádný speciální software, protože se Figma spouští přímo ve webovém prohlížeči. To usnadňuje sdílení projektů s klienty, kteří nemusí mít nainstalovaný speciální software.

Celkově lze říci, že Figma představuje efektivní a moderní nástroj pro designování digitálních produktů, který podporuje týmovou spolupráci a usnadňuje iterativní proces vývoje uživatelského rozhraní. *(DRAW PLANET, 2019)*

2.2 HTML

HTML, zkratka pro HyperText Markup Language (HyperTextový značkovací jazyk), je základní stavební kámen pro vytváření webových stránek a webových aplikací. Jedná se o standardizovaný značkovací jazyk, který umožňuje strukturovat obsah webových stránek a definovat vztahy mezi různými částmi.

HTML vytváří strukturu dokumentu pomocí tzv. HTML značek (tagů), které jsou uzavřeny do ostrých závorek. Tyto značky mohou obsahovat různé atributy, které poskytují dodatečné informace o obsahu. Každá HTML stránka začíná a končí značkami <html> a </html>, uvnitř kterých jsou další značky, jako například <head> (obsahující informace o dokumentu, jako jsou odkazy na styly, skripty atd.) a <body> (obsahující samotný viditelný obsah stránky). *(McFedries, 2019)*

Základními stavebními bloky HTML jsou například:

- <h1>, <h2>, ..., <h6>: Značky pro nadpisy různé úrovně.
- <p>: Značka pro odstavec textu.
- <a>: Značka pro odkazy.

- <img>: Značka pro vkládání obrázků.
- a <ol>: Značky pro nečíslovaný a číslovaný seznam.

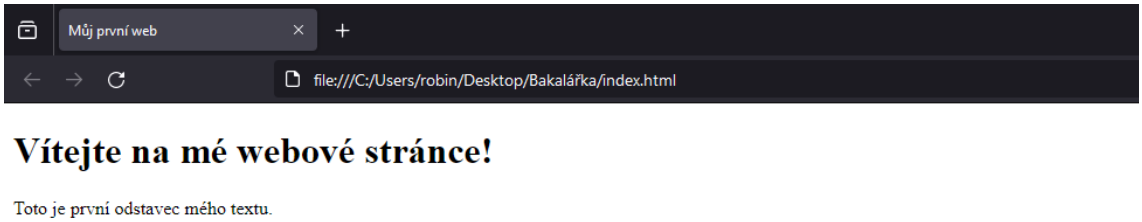
Příklad jednoduchého HTML dokumentu:

```
<!DOCTYPE html>
<html>
<head>
    <title>Můj první web</title>
</head>
<body>
    <h1> Vítejte na mé webové stránce! </h1>
    <p>Toto je první odstavec mého textu. </p>
    <p> <a href="https://www.example.com“> Odkaz na jinou
stránku</a> </p>
</body>
</html>
```

Obr. 3: Ukázka HTML

Zdroj: Vlastní

Pokud bychom tento kód vložili do souboru s příponou „.html“ a soubor spustili, objevila by se nám stránka s tímto obsahem.



Obr. 4: Ukázka webu

Zdroj: Vlastní

Do tagu „<head>“ dále můžeme připojovat kódy JavaScriptu (skripty) a tagy „<meta>“, které nám poskytují metainformace o webové stránce. Ty jsou důležité pro webové prohlížeče.

2.3 CSS

je stylovací jazyk používaný pro definování vzhledu a formátování webových stránek napsaných v HTML nebo jiných značkovacích jazycích. CSS umožňuje oddělit prezentaci (vzhled) od obsahu (strukturního obsahu), díky tomu poskytuje větší flexibilitu a údržbu vývoje webových stránek.

V kontextu odstavců může být CSS použito k nastavení různých stylů pro text v odstavcích, jako jsou barvy, velikosti písma, odsazení, mezery mezi řádky a mnoho dalších vlastností. CSS se aplikuje na stránky pomocí několika způsobů. Tři nejhlavnější jsou: Vnější CSS soubor, Interní CSS a Inline CSS.

Ukázka Inline CSS:

```
<p style="color: blue;"> Toto je první odstavec mého textu. </p>
```

Obr. 5: Ukázka CSS

Zdroj: Vlastní

Do tagu „<p>“ jsme přidali atribut „style“ a definovali pro něj „color: blue“, což náš text změní na modrou barvu.

CSS disponuje širokou škálou vlastností, které můžeme přidat různým prvkům a vytvářet tak profesionalitu webových stránek. Některé z klíčových vlastností CSS zahrnují možnost nastavení pozadí, průhlednosti, efektů přechodů, zakřivení rohů a mnoho dalších. Díky těmto vlastnostem lze dosáhnout komplexního a esteticky přitažlivého designu. *(McFedries, 2019)*

Pomocí CSS můžeme také získat responzivní design, což znamená, že webové stránky se automaticky přizpůsobí různým zařízením a velikostem obrazovek. To je klíčové pro zajištění optimální uživatelské zkušenosti bez ohledu na to, zda jsou stránky prohlíženy na počítači, tabletu nebo mobilním zařízení. *(McFedries, 2019)*

2.3.1 SCSS

SCSS (Sassy CSS) je nadmnožina syntakticky rozšířeného stylovacího jazyka CSS. Jedná se o metalanguage nad standardním CSS, což znamená, že všechny platné CSS kódy jsou také platné v SCSS. SCSS přidává několik vylepšení a nových funkcí, které usnadňují a zpřehledňují jak psaní, tak údržbu kaskádových stylů. *(Sass, 2023)*

Hlavní rozdíl mezi SCSS a klasickým CSS spočívá ve způsobu zápisu. SCSS používá syntaxi podobnou programovacím jazykům, což umožňuje vnořování pravidel, používání proměnných, funkcí a dalších pokročilých funkcí. To vše přispívá k větší modularitě a čitelnosti kódu. Funkce, které se vyskytují pouze v SCSS:

- Proměnné – umožnuje nám deklarovat a používat hodnoty, které se dají použít opakovaně v našem kódu.
- Vnořená pravidla – umožňuje vnořovat pravidla, a tak zlepšit čitelnost kódu.
- Mixiny – umožňuje vytvoření opakovaně použitelného kódu, čímž zlepšíme údržbu kódu.

- Importování souborů – umožňuje importovat další soubory SCSS do dalších a tím rozdělit kód pro lepší organizaci a údržbu.

2.4 JavaScript

JavaScript je programovací jazyk, který běží ve všech moderních prohlížečích. Je dynamicky typovaný, takže nemusíme při každé nové proměnné deklarovat její typ. Patří mezi vysokoúrovňové programovací jazyky, což znamená, že poskytuje abstrakce a úrovně abstrakce, které umožňují programátorům psát kód na vyšší úrovni, což usnadňuje porozumění a snižuje potřebu detailněji řešit strojové instrukce nebo správu paměti.

JavaScript je implementací standardu ECMAScript, což je standardní skriptovací jazyk. ECMAScript poskytuje specifikaci pro skriptovací jazyky a slouží jako základní norma pro jazyk JavaScript. V praxi jsou termíny "JavaScript" a "ECMAScript" často používány synonymicky, ale je důležité si uvědomit, že JavaScript je konkrétní implementací standardu ECMAScript.

ECMAScript je pravidelně aktualizován, přičemž nové verze přinášejí nové funkce, vylepšení a jazykové konstrukce. Například, ECMAScript 6 (ES6), také známý jako ECMAScript 2015, byl významnou aktualizací, která přidala mnoho nových možností do jazyka, včetně šablonových literálů, arrow funkcí, tříd a destrukturalizace. *(Flanagan, 2020)*

Tento programovací jazyk nám dává možnost přidávat různé funkce do naší stránky. Tím může být kliknutí na nějaké tlačítko, načítání dat z databáze, změna obsahu na stránce a plno dalších funkcí. Pro následující téma ukážu pouze pár prvků z tohoto jazyka.

```
document.getElementsByClassName('trida');
```

Řádek tohoto kódu nám vybere elementy pod názvem třídy „trida“. Existuje samozřejmě více způsobů, jak vybrat elementy a dále s nimi pracovat.

```
function UkazCislo(cislo) {
    console.log(cislo);
}
UkazCislo(5)
```

Obr. 6: JavaScript Funkce

Zdroj: Vlastní

Výše uvedený kód je ukázkou funkce v JavaScriptu. Funkce je blok kódu, který má za úkol vykonávat nějakou úlohu. Funkci musíme nejdříve definovat, to uděláme tak, že napíšeme klíčové slovo „function“ a za ním název naší funkce se závorkami na konci. V tomto případě to je „UkazCislo“. V závorkách se vyskytují takzvané „parametry“ funkce. Ty můžeme libovolně přidávat podle potřeby a podle toho, co chceme, aby naše funkce dělala. Zde předáváme argument „cislo“ a to dále vypíšeme do konzole. Poté tu funkci stačí zavolat, to provedeme tak, že napíšeme název funkce a přidáme do ní potřebné argumenty.

2.5 Asynchronní JavaScript

JavaScript je jednovláknový programovací jazyk. Existuje pouze jedno hlavní vlákno pro vykonávání kódu, to znamená, že se kód vykonává sekvenčně, jeden příkaz po druhém. Pokud bychom zapnuli program, který musí vykonat práci po dobu pěti vteřin, zablokoval by další kód, jenž následuje po něm a spustil by se po pěti vteřinách. Kód, který následuje po této události, by se nevykonal a stránka by se zdála být „zamrzlá*". (Haverbeke, 2018)*

Pro vyřešení tohoto problému tu máme Asynchronní JavaScript. Asynchronní JavaScript nám dovoluje, aby mohlo běžet více událostí najednou, a my tak nemuseli zbytečně čekat. Jedním z příkladu může být událost kliknutí. *(Haverbeke, 2018)*

```
document.getElementByID('tlacitko').addEventListener('click', ()
=> {
   console.log(1)
});
```

Obr. 7: Událost kliknutí

Zdroj: Vlastní

Tento kód nám na naše tlačítko s identifikátorem „tlacitko" přidává událost kliknutí. Pokaždé, když uživatel klikne na tlačítko, objeví se v konzoli číslo 1. Číslo jedna se neobjeví hned po spuštění kódu, ale pouze, pokud klikneme na tlačítko. Toto je jednoduchý příklad Asynchronního JavaScriptu. Moderní JavaScript disponuje různorodým spektrem asynchronních funkcí a jsou to Callback funkce, Promises a syntaxe async/await.

Pokud máme funkci, která trvá několik sekund, přidáme do argumentů další funkci která se spustí pouze v případě, že se původní funkce dokončí. Této funkci se říká „callback", v překladu „zpětné volání". *(Haverbeke, 2018)*

Ukázka callback funkce:

```
function ZiskaniDat(callback) {
    console.log("Získávám data 5 sekund...");
    setTimeout (function () {
        callback("Operace dokončena!");
    }, 5000);
}
```

Obr. 8: Callback funkce

Zdroj: Vlastní

Pokud tuto funkci zavoláme, vypíše se do konzole „Získávám data 5 sekund…", poté se spustí časovač pro 5 sekund a po těchto 5 sekundách se zavolá funkce „callback", která vypíše do konzole, že se operace dokončila. Funkce SetTimeout je pouze simulací čekání na získání dat.

Druhé možné řešení je pomocí Promisů. Promise je dalším z asynchronních zápisů, které můžeme použít k řešení našich problémů s blokováním kódu, čekání na načtení dat a další. Promise nám vrací objekt, který může být k dispozici ihned, později, anebo nikdy. Je to operace, která nemusí být okamžitě dokončena a může trvat nějaký čas. *(Haverbeke, 2018)*

Má tři vlastnosti:

1. **Stav (State):** Promise může být ve třech stavech:
   - **Pending (čekající):** Po vytvoření promise je v počátečním stavu, kdy je asynchronní operace stále neukončena.
   - **Fulfilled (splněná):** Operace byla úspěšně dokončena a promise obsahuje výslednou hodnotu.
   - **Rejected (odmítnutá):** Operace selhala a promise obsahuje chybovou informaci.
2. **Hodnota (Value):** Jakmile je promise splněna nebo odmítnuta, obsahuje hodnotu, která reprezentuje výsledek asynchronní operace.
3. **Funkce (Handlers):** Promise umožňuje přidání tzv. "handlers" nebo "callbacks," které budou provedeny po splnění nebo odmítnutí promise. Tyto funkce mohou manipulovat výsledek asynchronní operace.

Jednoduchá ukázka Promisu, kterému ihned definujeme splněný stav:

```
let cislo = Promise.resolve(10);

cislo.then(cislo => console.log("Moje číslo je ${cislo}");
```

Obr. 9: Promise
Zdroj: Vlastní

„Promise.resolve“ nám umožňuje ihned vložit hodnotu do promisu a voláním funkce „then“, získáme vloženou hodnotu. Ve výše uvedeném kódu se nám do konzole vypíše „Moje číslo je 10“. Pokud by se stal problém a data by nemohla být získána, můžeme použít volání funkce „catch“, která nám zachytí odmítnutý stav promisu , a tak s ním můžeme pracovat dále. *(Haverbeke, 2018)*

Třetím a posledním způsobem, jak psát asynchronní kód, je pomocí syntaxe async/await. Jedná se o jednodušší a čitelnější zápis asynchronní funkce. Tuto funkci vytvoříme pouze, když před naší klasickou funkci přidáme klíčové slovo „async“. To nám dovolí použít v této funkci klíčové slovo „await“. Await čeká na vyhodnocení Promise a uloží do výsledku jeho hodnotu, s kterou můžeme v budoucnu pracovat.

```
async function AsynchronniFunkce() {
  const vysledek = await nejakyPromise();
}
```

Obr. 10: Async/Await
Zdroj: Vlastní

Do „vysledek“ se nám uloží výsledek z funkce „nejakyPromise“, na kterou čekáme pomocí klíčového slova await. Zde bychom samozřejmě mohli přidat také „catch“ pro zachycení chybového stavu. *(Haverbeke, 2018)*

2.6 PHP

PHP je skriptovací programovací jazyk navržený především pro vývoj webových stránek. Jedná se o back-end technologii, tedy o serverový jazyk, což znamená, že kód PHP běží na straně serveru a generuje dynamický (měnící) obsah, který je poté poslán klientovi. PHP je široce využíváno ve světě webového vývoje a často se používá ve spojení s databázovými systémy, jako je MySQL, HTML, CSS, a JavaScript.

Popularita PHP vyplývá také z jeho jednoduchosti použití a rozsáhlé komunity, která poskytuje bohatou dokumentaci a mnoho dostupných knihoven a frameworků pro vývoj. Ve své práci ho používám hlavně pro práci s databází. *(Davis a Phillips, 2007)*

2.7 Visual Studio Code

Visual Studio Code je bezplatný open-source textový editor vyvinutý a udržovaný společností Microsoft. Je určen pro vývojáře a podporuje mnoho programovacích jazyků. Několik klíčových rysů Visual Studio Code zahrnuje:

- Rozšíření a pluginy: VSCode má rozsáhlý ekosystém rozšíření a pluginů, které umožňují vývojářům přizpůsobit editor podle svých potřeb. Tato rozšíření zahrnují podporu pro různé programovací jazyky, nástroje pro správu verzí, linters, debuggery a mnoho dalšího.
- Integrovaný správce verzí: VSCode má integrovaný správce verzí, což usnadňuje sledování změn ve vašem kódu a spolupráci s dalšími vývojáři.
- Integrovaný debugger: Podporuje ladění kódu přímo v editoru a umožňuje nastavit zástavby, sledovat proměnné a další debuggovací funkce.
- Inteligentní automatické doplňování: Nabízí chytré automatické doplňování kódu a funkce zvyšující produktivitu vývojářů.
- Nízké nároky na systém: Je navržen tak, aby byl lehký a rychlý. Může běžet na různých operačních systémech, včetně Windows, macOS a Linux.

Visual Studio Code obsahuje integrovaný terminál, který nám umožňuje spouštět příkazy přímo v editoru. Toto využijeme zejména pro následující technologii npm.

2.8 Npm

Npm je zkratka pro "Node Package Manager" a je to správce balíčků pro platformu Node.js. Node.js je open-source, cross-platform JavaScriptový běhový prostředek, který umožňuje spouštění JavaScriptového kódu na straně serveru. Npm je nástroj, který umožňuje vývojářům instalovat, spravovat a sdílet balíčky (kódy a knihovny napsané v jazyce JavaScript), ty mohou být použity ve svých projektech na platformě Node.js. Můžeme ho použít i ve front-end vývoji.

NPM umožňuje vývojářům snadný přístup k tisícům balíčků a knihoven, které jsou k dispozici v repozitáři npm. Vývojáři mohou pomocí příkazů jako npm install nainstalovat balíčky do svých projektů a npm publish umožňuje sdílet vlastní balíčky s komunitou.

NPM hraje klíčovou roli ve světě JavaScriptového vývoje, je také široce používán pro správu závislostí a balíčků ve mnoha typech projektů.

2.9 MySQL

Bezplatná, ale plnohodnotná relační databáze SQL byla vyvinuta v 90. letech 20. století a slouží k inteligentnímu uchovávání a využívání dat nebo informací. Používá se zejména ve spojení s webovými aplikacemi.

Relační databáze znamená, že data jsou organizována do tabule se vztahy mezi nimi, a to umožňuje efektivní organizaci a práci s daty.

MySQL je distribuován pod open-source licencí, což znamená, že jeho zdrojový kód je volně dostupný a může být upravován podle potřeb uživatelů.

Používá standardní jazyk SQL (Structured Query Language) pro definici, manipulaci a dotazování dat v databázi. To znamená, že uživatelé mohou používat standardní SQL dotazy pro práci s MySQL databází. *(Davis a Phillips, 2007)*

Databáze je navržena tak, aby nabízela vysoký výkon a efektivitu. Je schopna zpracovávat obrovské objemy dat a je používána v různých odvětvích, od malých webových stránek po velké webové aplikace.

2.10 Webpack

Webpack je nástroj pro balení (bundling) a správu závislostí v moderním webovém vývoji. Jeho hlavním účelem je optimalizovat a sjednotit způsob, jakým webové aplikace zpracovávají a distribuují svůj kód a závislosti. Když webpack sestavuje naší aplikaci, interně sestaví graf závislostí z jednoho nebo více vstupních bodů a poté spojí všechny moduly do jednoho nebo více svazků.

Balení (bundling) je spojování dvou a více souborů s kódem a dalších zdrojů do jednoho nebo více balíčků. Tíme se snižuje počet http požadavků a zlepšuje se efektivita načítání webové stránky.

Webpack podporuje modulární programování, což umožňuje vývojářům rozdělit kód do menších, opakovaně použitelných částí. To zlepšuje udržovatelnost a znuvupoužitelnost kódu.

Umožňuje nám zpracovávání různých typů souborů a asettů, jako jsou styly, obrázky, písma a podobně. S využitím takzvaných „loaderů“ může transformovat tyto soubory do formátů vhodných pro webové použití.

Nabízí nám širokou škálu pluginů pro optimalizaci. Pluginy nám pomáhají v optimalizaci kódu, minifikaci, odstranění nepotřebných částí, generování manifestů a další úpravy.

Abychom mohli Webpack používat v našich projektech, je nutné pochopit pár klíčových konceptů a nimi jsou:

- Entry (Vstup)
- Output (Výstup)
- Loaders (Načítací modul)
- Plugins (Pluginy)
- Mode (Mód)
- Browser Compatibility (Kompatibilita prohlížečů)

Entry, neboli vstupní bod, indikuje, který modul bude začátkem našeho interního grafu závislostí. Webpack poté zjistí, jaké další moduly a prostředky jsou závislé na tomto bodě. Tyto body se nastavují v konfiguračním souboru webpacku, a to tímto způsobem:

```
module.exports = {
  entry: 'cesta/k/memu/souboru.js',
};
```

Obr. 11: Webpack Entry

Zdroj: Vlastní

Uvedený kód je součástí konfiguračního souboru webpacku a definují se v něm různé funkce, které potřebujeme ke své práci. Do vstupního bodu (entry) se zapisuje cesta k našemu javascriptovému souboru a tím zajistí, že všechno, co má nějakou závislost s naším souborem, bude propojeno, zabaleno a optimalizováno do jednoho výstupního souboru.

Dalším důležitým bodem v konceptu Webpacku je správa výstupu, která určuje, kam a jakým způsobem budou optimalizované a propojené soubory uloženy. V konfiguraci Webpacku je možné nastavit různé parametry výstupu.

```
module.exports = {
entry: 'cesta/k/memu/souboru.js',
output: {
filename: 'VystupniSoubor.js',
  },
};
```

Obr. 12: Webpack Output

Zdroj: Vlastní

Zde aplikujeme výstupní bod pomocí položky „output“ a v ní specifikujeme, že chceme, aby se náš výstupní soubor jmenoval „VystupniSoubor“.

Koncept načítacích modulů slouží k načítání různých souborů. Webpack rozumí jen javascriptovému a JSON souboru, a proto potřebujeme načítací moduly, abychom mohli načíst písma, obrázky, css soubory a další. Pokud tyto načítací moduly definujeme, Webpack zahrne do našeho grafu závislostí soubory, které chceme načíst. Má dvě vlastnosti, těmi jsou test a use. Vlastnost test identifikuje, který soubor nebo soubory mají být transformovány, jinak řečeno, které Webpack hledá pro načítání. Tyto soubory definujeme jejich příponami .txt, .jpg, .ttf a ostatní. Píše se ve formě regulárních výrazů. Use ukazuje, který načítací modul by měl být použit k provedení načtení. Pro obrázky potřebujeme například načítací modul s názvem file-loader.

```
module.exports = {
  output: {
    filename: 'mujPrvniZabalenySoubor.js',
  },
  module: {
    rules: [{ test: /\.txt$/, use: 'raw-loader' }],
  },
};
```

Obr. 13: Webpack Moduly

Zdroj: Vlastní

V tomto případě testujeme soubory s příponout .txt, tedy textové soubory a načítáme je pomocí raw-loader pro textové soubory.

Pluginy jsou speciální typy rozšíření, které umožňují provádět širokou škálu úloh během procesu balení. Tyto pluginy poskytují možnost optimalizovat kód, manipulovat soubory, a provádět mnoho dalších operací.

```
const HtmlWebpackPlugin = require('html-webpack-plugin');
const webpack = require('webpack');
module.exports = {
  module: {
    rules: [{ test: /\.txt$/, use: 'raw-loader' }],
  },
  plugins: [new HtmlWebpackPlugin({ template: './src/index.html'
})],
};
```

Obr. 14: Webpack Pluginy

Zdroj: Vlastní

Ve výše uvedeném kódu implementujeme plugin html-webpack-plugin, který nám generuje HTML soubor a automaticky do něho vkládá všechny soubory, které jsou závislé s tímto HTML souborem.

Mód určuje, v jakém režimu by měl Webpack pracovat. Volba mode přijímá jeden ze tří režimů, těmi jsou production, development nebo none. V režimu development jsou aktivovány různé nástroje pro usnadnění ladění a analýzu kódu. Zpravidla se vytváří neoptimalizovaný kód s ohledem na rychlou sestavu a snadnou čitelnost. Zahrnuje informace pro ladění v konzoli a zachovává původní strukturu souborů pro snazší odhalení chyb. V režimu production je kód optimalizován pro výkon a velikost. Jsou použity optimalizace, jako je minifikace kódu, odstranění nepoužívaných částí kódu a další postupy pro zmenšení velikosti výsledného balíku.

Režim none deaktivuje všechny přednastavené optimalizační možnosti a nastavení související s režimem. Vývojář může ručně konfigurovat, které optimalizace nebo nástroje chce použít.

Musíme také dbát na kompatibilitu s webovými prohlížeči, neboť Webpack podporuje všechny prohlížeče, které jsou v souladu s ES-5. Pro správnou funkcionalitu Webpacku je zapotřebí podpora Promise pro klíčová slova pro import nebo připojení souborů. Pokud plánujeme podporovat i starší prohlížeče, budeme muset použít polyfill. Polyfill je kód, který implementuje funkce, jež nejsou přítomné v daném prohlížeči, a zajistí tak potřebnou kompatibilitu.

2.11 API

API neboli rozhraní pro programování aplikací, je klíčovým prvkem moderního softwarového vývoje. Jedná se o soubor definicí a pravidel, které usnadňují komunikaci mezi různými softwarovými částmi, a to jak uvnitř jedné aplikace, tak i mezi různými aplikacemi.

Hlavním cílem API je umožnit integraci a interoperabilitu mezi různými systémy. Když mluvíme o API v kontextu databází, obvykle máme na mysli možnost posílat a získávat data z databáze. API databáze poskytuje sadu funkcí a procedur, které mohou být volány z jiných částí softwaru.

API definuje, která data jsou akceptována a jak mají být prezentována. To zahrnuje formáty dat, jako jsou JSON nebo XML. Dále specifikuje protokoly komunikace, například používání HTTP při komunikaci mezi klientem a serverem. Bezpečnostní aspekty jsou také klíčovou součástí API, včetně autentizace (ověření identity) a autorizace (kontroly oprávnění k přístupu k datům).

Celkově lze říci, že API je mostem, který umožňuje propojení a spolupráci mezi různými částmi softwaru, a tím zvyšuje flexibilitu a efektivitu softwarových systémů. API, které je využíváno v této práci, je fetch.

Fetch API představuje moderní způsob, jak v jazyce JavaScript provádět HTTP požadavky, poskytující vylepšený a jednoduchý přístup oproti staršímu XMLHttpRequest (XHR). Jeho hlavním přínosem je snadná čitelnost a promise-based model, což usnadňuje práci s asynchronními operacemi.

Využívání Fetch API je intuitivní. Při odesílání požadavku je vytvářen objekt typu Request, který lze detailně konfigurovat, například stanovením metody (GET, POST) či přidáním vlastních hlaviček. Po odeslání požadavku Fetch API vrací objekt Response, který obsahuje informace o odpovědi ze serveru.

Tento model odpovědi zahrnuje nejen samotná data, ale také metadata, jako jsou status kód, hlavičky a další informace. Fetch API dále podporuje streamování dat, což je užitečné zejména při práci s velkými soubory nebo když je nutné postupně načítat obsah.

Následující kódový úryvek představuje jednoduchý použití Fetch API pro získání dat z určené URL:

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Chyba:', error));
```

Obr. 15: Fetch API

Zdroj: Vlastní

Tento kód provádí GET požadavek na zadanou URL, zpracovává odpověď ve formátu JSON a výsledek vypisuje do konzole. Celkově Fetch API poskytuje moderní a výkonné nástroje pro manipulaci s HTTP požadavky v rámci webových aplikací. *(Mozilla Corporation, 2023)*

2.12 Základní elementy k vytvoření stránky

Předtím, než budeme navrhovat (designovat) webovou stránku, musíme si ustanovit základní body, bez nichž by se proces vývoje neuskutečnil. Těmito elementy jsou webový hosting, doménové jméno a vývojové metody, jinak řečeno, jaké nástroje budeme používat při vytváření naší webové aplikace.

2.12.1 Webhosting

Webhosting je služba, která umožňuje jednotlivcům nebo organizacím zveřejňovat své webové stránky nebo webové aplikace na internet. Poskytovatel této služby nám poskytne místo na jeho serveru a nabízí nám také další funkce, které můžeme z pozice zákazníka využít. To mohou být služby jako je databáze, uložiště, přenos dat, doménové jméno, e-mailové služby, zabezpečení a technická podpora. Webhosting nám také umožní, aby naše stránky byly dostupné 24 hodin denně a s kvalitní rychlostí.

Webhosting patří mezi nejlevnější způsoby, jak vložit stránky na internet. Ušetří nám to spoustu času a nákladů, protože nemusíme pořizovat vlastní server a konfigurovat ho.

Na webhostingu se ukládá:

1. Databáze souborů (obrázky, hudba, text a další)
2. CMS – nainstalovaný redakční systém, který pro tvorbu svého webu používáte (WordPress, Joomla, Drupal apod.).
3. HTML, CSS, Javascript, PHP soubory, v nichž je uložený programovací kód vašeho webu.
4. E-maily (máte-li na své doméně zřízenou e-mailovou schránku – tedy když místo např. obchod@gmail.com používáte obchod@mojedomena.cz).

2.12.2 Doménové jméno

Doménové jméno (doménu) můžeme chápat jako adresu naší webové stránky, pod kterou nás klienti mohou dohledat, například www.seznam.cz. Tato jména jsou součástí systému DNS, tedy Domain Name Systém, který tato jména převádí na IP adresy, jež jsou skutečně používány pro identifikaci jednotlivých serverů na internetu. Struktura doménového jména se skládá z hlavní části. To je část za poslední tečkou .com, .cz a podobně. Druhá část představuje

unikátní název, který si uživatelé mohou vybrat. V příkladu example.com je „example“ druhá část domény.

2.12.3 Vývojové nástroje

Vývojové nástroje nám pomáhají s vytvořením naší stránky. Jak už již bylo zmíněno, tyto nástroje jsou programovací jazyky, frameworky, preprocesory a ostatní, které uznáme za vhodné. Pomocí těchto nástrojů poté můžeme realizovat naší webovou aplikaci nebo stránku.

2.12.4 Optimalizace pro vyhledávače

SEO znamená "Search Engine Optimization" neboli optimalizace pro vyhledávače. Jedná se o soubor postupů a strategií, které mají za cíl zlepšit viditelnost a pozici webových stránek ve výsledcích vyhledávačů (např. Google, Bing) při relevantních dotazech uživatelů.

Klíčové Aspekty SEO:

- Klíčová Slova (Keywords) - Identifikace klíčových slov a frází relevantních pro obsah webových stránek a vyhledávací dotazy uživatelů.
- On-Page SEO – Optimalizace obsahu webových stránek, včetně nadpisů, meta popisů, URL adres, obrázků a dalších prvků přímo na stránkách.
- Technické SEO – Zlepšení technických aspektů webových stránek, jako jsou rychlost načítání, mobilní optimalizace, používání správných HTML značek a další.
- Off-Page SEO – Budování autority stránek skrze zpětné odkazy (backlinky) od jiných důvěryhodných a relevantních webových stránek.
- Lokální SEO – Optimalizace pro místní vyhledávání, zejména pro podniky s fyzickou přítomností v určité oblasti.
- Obsahový Marketing – Vytváření kvalitního a hodnotného obsahu, který zaujme uživatele a odpovídá jejich potřebám.
- Analýza a Měření – Sledování výsledků, analýza návštěvnosti, a pravidelné měření úspěšnosti optimalizačních strategií.
- Sociální Média – Aktivní účast na sociálních médiích může přispět ke zlepšení viditelnosti obsahu a návštěvnosti.
- UX (User Experience) - Zajištění pozitivního uživatelského zážitku pro návštěvníky stránek.

Cílem SEO je získat co nejlepší pozice ve výsledcích vyhledávání pro relevantní klíčová slova, což může zvýšit návštěvnost webových stránek. SEO je dynamickým procesem, který vyžaduje neustálou optimalizaci a sledování měnících se algoritmů vyhledávačů. *(Third Door Media, 2023)*

2.13 Optimalizace webových stránek

V digitálním věku, kde se internet stal nepostradatelnou součástí našeho každodenního života, je klíčové zajistit, aby webové stránky poskytovaly uživatelům co nejlepší možnou zkušenost. Optimalizace webových stránek se stává nezbytným prvkem úspěšné online přítomnosti. Tato strategie nejenže podporuje viditelnost na internetu, ale také přispívá ke zlepšení uživatelské interakce.

Optimalizace webových stránek se zabývá širokým spektrem faktorů, které ovlivňují efektivitu a účinnost online prostoru. To zahrnuje nejen obsah a strukturu stránek, ale také navigaci, uživatelskou přívětivost a mnoho dalšího. Cílem je vytvořit prostředí, které je nejen atraktivní pro návštěvníky, ale také efektivní při plnění cílů daného online projektu.

V následujícím tématu se zaměříme na různé aspekty optimalizace webových stránek, od designu až po obsahovou strategii Budou rozebírány klíčové postupy a metody, které pomáhají dosáhnout maximálního potenciálu webové stránky. Bez ohledu na to, zda jste jednotlivec usilující o lepší viditelnost, nebo firma hledající efektivní online platformu. Optimalizace webových stránek je klíčem k úspěchu v digitálním prostoru.

2.13.1 Rychlost načítání

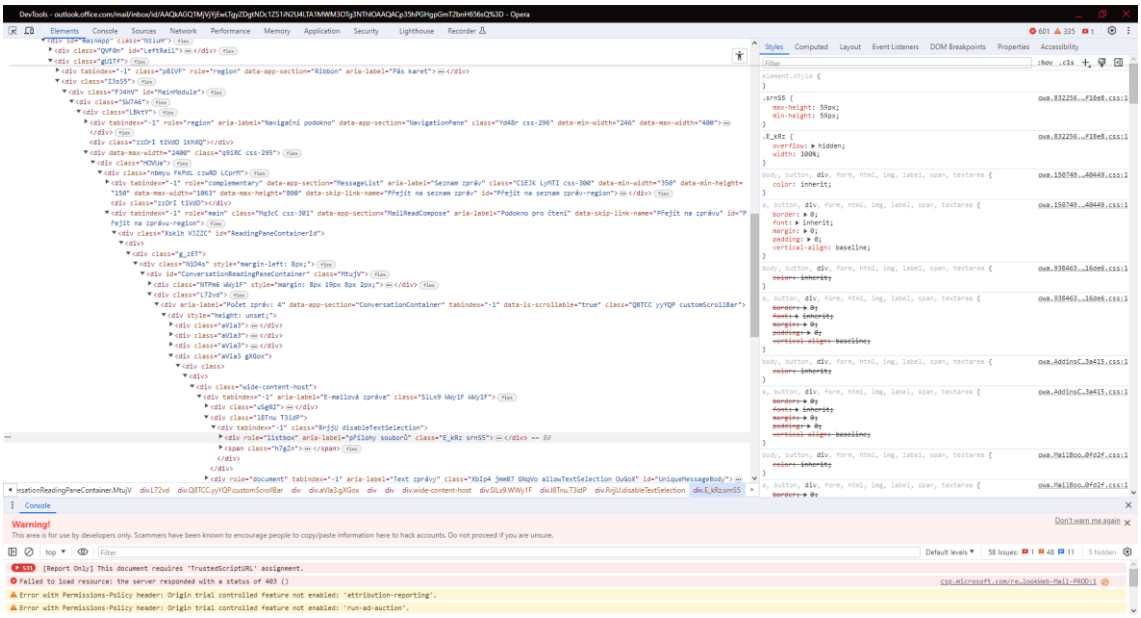
Rychlost načítání webových stránek představuje klíčový aspekt, který může značně ovlivnit uživatelskou spokojenost a úspěch online projektu. Základním principem je doba, kterou stránka potřebuje k zobrazení obsahu po zadání URL nebo provedení interakce. Tato doba má významný vliv na to, zda uživatelé na stránce zůstanou, nebo se rozhodnou ji opustit.

Rychlost načítání není pouze estetickým prvkem, ale má také důsledky pro optimalizaci pro vyhledávače (SEO). Hledací algoritmy, zejména v případě Google, hodnotí rychlé stránky pozitivně, což může přinést výhodu ve výsledcích vyhledávání.

Existuje několik faktorů, které mohou ovlivnit rychlost načítání. Velké soubory obrázků mohou zpomalit načítání, a proto je důležité zajistit jejich efektivní optimalizaci a kompresi. Redukce nadbytečného kódu, minifikace a shlukování CSS a JavaScript souborů jsou další klíčové strategie pro snížení objemu dat, jež musí být stahována.

Pro měření a optimalizaci rychlosti načítání existuje několik nástrojů. Google PageSpeed Insights, Lighthouse a GTmetrix poskytují podrobné analýzy výkonu stránek a nabízejí doporučení pro vylepšení.

V mém konkrétním případě jsou využívány nástroje pro vývojáře, které jsou dostupné v každém prohlížeči. Pokud chceme nahlédnout do těchto nástrojů, stačí když si otevřeme nějakou webovou stránku a stiskneme klávesovou zkratku Ctrl+Shift+I. Po stisknutí se otevřou vývojářské nástroje.



Obr. 16: DevTools

Zdroj: Vlastní

V záložce "Prvky" mohou vývojáři detailně prozkoumat strukturu HTML a CSS jednotlivých prvků na stránce. Tato funkce umožňuje dynamické úpravy, což je užitečné pro rychlé testování změn v designu nebo rozložení obsahu.

Síťový panel poskytuje důležitý přehled o síťových požadavcích provedených během načítání stránky. Vývojáři mohou sledovat časy odezvy, velikosti přenesených souborů, a identifikovat potenciální problémy, které by mohly zpomalovat načítání stránky.

Záložka Konzole slouží k sledování chyb v JavaScriptu a umožňuje vývojářům interaktivně spouštět příkazy. To je klíčové pro ladění a rychlou diagnostiku potenciálních problémů ve skriptech.

Aplikační panel zase nabízí informace o uložených datech, cookies a offline datovém ukládání. To je užitečné při vývoji webových aplikací, které využívají lokální úložiště.

Celkově vzato, umožňují tyto vývojářské nástroje nejen vytvářet a upravovat kód, ale též poskytují prostředky pro rychlé a efektivní řešení problémů, což je důležité pro moderní vývoj webových aplikací.

2.13.2 Optimalizace obrázků

Optimalizace obrázků je klíčovým prvkem v procesu vytváření webových stránek, který se zaměřuje na efektivní správu grafických prvků s cílem maximalizovat rychlost načítání a celkový výkon stránky. Jeden z klíčových rozhodujících faktorů je volba vhodného formátu pro jednotlivé obrázky. Například formát JPEG, vynikající pro fotografie s vysokým rozlišením, nabízí efektivní kompresi s minimální ztrátou kvality. Naopak PNG se často používá pro obrázky vyžadující průhlednost, a SVG, jako vektorový formát, se osvědčuje pro ikony a grafiku, kterou je třeba upravovat bez ztráty kvality.

Kromě volby formátu hraje klíčovou roli i správná aplikace kompresních algoritmů. Efektivní komprese umožňuje zmenšit velikost souborů bez významné ztráty kvality obrazu. To je zvláště důležité v době, kdy uživatelé očekávají rychlé načítání webových stránek. Existuje široká škála online nástrojů, které nám umožní kompresi obrázků, jeden z nich může být Optimizilla.

Optimalizace obrázků není pouze technickým úkolem, má také významný dopad na celkový uživatelský zážitek. Rychle načítající se stránky nejen zlepšují uživatelskou spokojenost, ale mohou také snížit míru odchodu návštěvníků.

2.13.3 Caching

Prohlížečový caching umožňuje ukládání lokálních kopií webových stránek a přidružených zdrojů (jako jsou obrázky, styly, nebo skripty) na zařízení uživatele. Tím se minimalizuje doba načítání při opakovaném navštívení stránky, neboť data mohou být okamžitě získána z mezipaměti zařízení.

Na úrovni serveru lze implementovat různé metody cachingu, například ukládáním kompletních HTML stránek nebo částí stránek. Toho lze dosáhnout prostřednictvím cache pamětí serverů, proxy serverů nebo obslužných vrstev, jako je populární nástroj Varnish.

K dalšímu řízení chování cachingu slouží HTTP cache hlavičky. Nastavením Expires nebo Cache-Control hlaviček lze definovat dobu platnosti cachovaných dat, což umožňuje efektivní správu obsahu v mezipaměti.

Zajímavým prvkem cachingu je i verze souborů. Přidáním verzí nebo hashů do názvů souborů umožňuje efektivně aktualizovat cachované soubory po jejich změnách, což zajistí konzistenci obsahu pro uživatele.

2.13.4 Správa skriptů a stylů

Optimalizace webových stránek zahrnuje účinnou manipulaci s JavaScriptovými skripty a styly CSS, což představuje klíčový prvek v dosažení rychlého načítání a optimalizace výkonu.

Minimalizace a sjednocení souborů skriptů a stylů je klíčovým krokem v této oblasti. Tímto způsobem je možné snížit objem dat, které uživatelé musí stáhnout při každém načítání stránky. Redukce počtu a velikosti souborů je často dosahována kombinováním a minimalizací kódu, což výrazně přispívá ke snížení času potřebného pro načítání stránky.

Dalším důležitým prvkem je asynchronní nebo odložené načítání skriptů. Asynchronní načítání umožňuje skriptům stahovat se nezávisle na načítání stránky, což minimalizuje čekací dobu pro uživatele. Odložené načítání je ideální pro skripty, které nejsou kritické pro počáteční vykreslení stránky, což může urychlit samotné načítání.

Z hlediska výkonu hraje roli také technika lazy loading, tedy odkládání načítání obrázků do chvíle, kdy jsou skutečně potřebné. Tím se snižuje initiační čas stránky a je umožněno rychlejší zobrazení klíčového obsahu.

2.13.5 Minifikace a shlukování kódu

Minifikace kódu představuje proces, během něhož je kód zkomprimován odstraněním bílých znaků, komentářů a jiných nepodstatných prvků. Výsledkem je výrazné zmenšení velikosti souborů, což má přímý dopad na rychlost přenosu dat při načítání stránky. Tímto způsobem můžeme dosáhnout optimalizace doby načítání a zároveň minimalizovat potřebné šířky pásma.

Dalším důležitým prvkem je shlukování kódu, což znamená kombinování více souborů kódu do jednoho balíčku. Tato technika snižuje počet HTTP požadavků, které je potřeba načíst stránku, což má za následek efektivnější načítání obsahu. Nástroje jako Webpack jsou běžně využívány k automatizaci tohoto procesu, umožňujícího komplexní správu závislostí a optimalizaci kódu.

```
//Normální kód
function Pozdrav(jmeno) {
    console.log("Ahoj, " + jmeno + "!");
}

Pozdrav("John");

//Minifikovaný kód
function Pozdrav(a){console.log("Hello, "+a+"!")}Pozdrav("John");
```

Obr. 17: Porovnání kódu
Zdroj: Vlastní

2.13.6 Mobilní optimalizace

Responzivní design je také klíčovou strategií v oblasti navrhování a vytváření webových stránek. Klade si za cíl automaticky přizpůsobit obsah různým zařízením a obrazovkovým velikostem. Jeho hlavním úkolem je poskytnout uživatelům optimální zážitek bez ohledu na to, zda používají počítač, tablet nebo mobilní telefon. V následujícím textu se blíže podíváme na klíčové principy responzivního designu.

Flexibilní mřížka je jedním ze základních prvků responzivního designu. Tato flexibilita umožňuje pružné rozložení prvků na stránce pomocí procentuálních hodnot nebo jednotek, jako jsou em a rem. Tímto způsobem se obsah snadno přizpůsobí různým obrazovkovým velikostem, což je klíčovým aspektem responzivního designu.

Dalším důležitým principem jsou pružné obrázky a média. Nastavením těchto prvků tak, aby se přizpůsobily velikosti zobrazovacího zařízení, zajišťujeme, že se budou měnit v souladu s velikostí okna prohlížeče. Například použitím CSS vlastnosti max-width: 100% zabraňujeme přetékání obrázků a zajišťujeme, že zůstanou v optimální velikosti.

Media Queries jsou soubory CSS pravidel, které umožňují aplikovat různé styly na různých zařízeních nebo obrazovkových velikostech. Tyto dotazy testují charakteristiky zařízení, jako je šířka obrazovky, orientace nebo rozlišení, což umožňuje přizpůsobit design stránky podle konkrétních potřeb uživatelů.

Další princip responzivního designu spočívá v používání relativních jednotek, například procent nebo em. Tímto způsobem jsou velikosti a odsazení definovány vzhledem k rodičovskému prvku nebo celkovému oknu prohlížeče, což přispívá k pružnému designu.

Viewport Meta Tag je klíčovým prvkem pro nastavení šířky a škálování stránky v prohlížeči. To zajistí, že obsah bude odpovídat velikosti obrazovky a nebude příliš malý nebo příliš velký.

Princip postupného zlepšení zdůrazňuje vytváření základních stránek, které fungují na všech zařízeních, a následně přidávání pokročilých funkcí nebo stylů pro modernější prohlížeče a zařízení. Tento přístup zajišťuje, že i uživatelé s méně moderními zařízeními mají přístup k základním funkcím.

V neposlední řadě je klíčovým principem testování na různých zařízeních. Důkladné testování webových stránek na různých zařízeních, od počítačů po tablety a mobilní telefony, je nezbytné pro zajištění konzistentního a kvalitního uživatelského zážitku napříč různými platformami. *(Google)*

2.14 Design

Design webových stránek zahrnuje celkovou koncepci a prvky, které ovlivňují nejen vizuální dojem, ale také funkčnost stránky. Pečlivý přístup k estetice a uživatelskému zážitku hraje klíčovou roli v úspěchu webového designu.

Estetika hraje důležitou roli při vytváření atraktivního vizuálního dojmu. Volba barev a harmonizace barevné palety jsou pečlivě vybírány s ohledem na cílovou skupinu a brand stránky. Správně zvolená typografie podporuje čitelnost a přehlednost textového obsahu.

Grafické prvky, jako jsou obrázky, ikony a grafika, jsou důležité pro zdůraznění klíčových informací a vytváření jedinečného vizuálního stylu. Jejich umístění a propojení s obsahem přispívá k uživatelsky přívětivému prostředí.

Celkový design by měl nejen vizuálně oslovovat, ale také efektivně fungovat. Zahrnuje rovnováhu mezi estetikou a funkčností, která napomáhá uživatelům snadněji interagovat s obsahem stránky.

2.14.1 UI/UX

UI (User Interface) a UX (User Experience) design jsou klíčovými součástmi vytváření efektivních webových stránek. Zatímco UI design se zaměřuje na vizuální aspekty a interakce, UX design zdůrazňuje uživatelskou zkušenost a celkovou pohodu při používání stránek.

UI design klade důraz na barevný design, typografii a grafické prvky. Pečlivý výběr barev a harmonická paleta přispívají k atraktivnímu vizuálnímu dojmu. Správná volba písma a jeho velikosti má významný dopad na čitelnost obsahu, a grafické prvky jsou klíčem k zvýraznění klíčových informací.

Na druhé straně UX design se zaměřuje na uživatelskou interakci a pohodlí. Logická struktura menu, optimalizace pro různá zařízení a interaktivní prvky podporují pohodlnou uživatelskou

zkušenost. Testování na skupinách uživatelů je nezbytnou součástí UX designu, aby se získala cenná zpětná vazba a mohly se provést potřebné úpravy.

3 Realizace webových stránek

Tato kapitola je věnována detailnějšímu popisu tvorby webových stránek. Cílem je vytvořit fungující web s rezervačním systémem, který bude vyhovovat požadavkům. Celé stránky jsou vytvořeny pomocí technologií, které již byly zmíněny.

3.1 Analýza současného webu

Prvním důležitým krokem je důkladná analýza současného stavu existujícího webového prostředí. Tato analýza nám poskytne ucelený pohled na stav, funkcionality a nedostatky současného webu, které je třeba řešit. Budeme se zabývat přehledem obsahu, uživatelským rozhraním, navigací, rychlostí načítání stránek a dalšími klíčovými aspekty, které ovlivňují uživatelský zážitek a efektivitu webového prostředí pro podporu podnikání.

3.2 Návrh designu

Na základě zjištění z analýzy současného stavu provedeme návrh nového designu s funkčností webové stránky. Zaměříme se na moderní designové prvky, které budou podporovat estetický dojem a zároveň vylepší uživatelskou přívětivost. Dále se budeme soustředit na jednoduchost webu, aby se v něm dalo jednoduše navigovat. Využijeme paletu barev, která odpovídá správnému kontrastu, kvůli starším návštěvníkům webu. Grafickým prvkem, který využijeme, budou i různé animace a efekty pro lepší a hezčí dynamiku stránky. Design se bude realizovat v programu Figma, který byl zmíněn v teoretické části. Design byl realizován od nuly, bez jakékoliv předlohy a inspirace.

3.2.1 Návrh domovské stránky a navigace

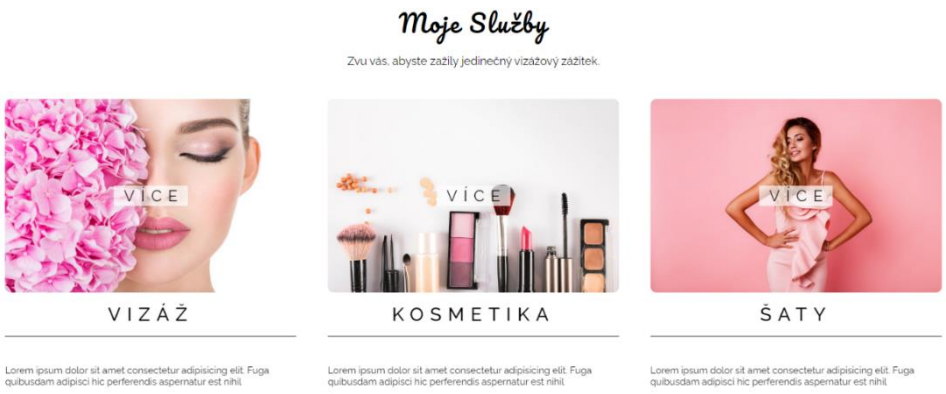
Domovská stránka se skládá z navigace, uvítacího banneru a fotografie v pozadí. Navigace obsahuje pět odkazů. Odkaz „domů“ slouží k návratu zpět na hlavní stránku. Služby nás přesměrují na stránku, kde se bude obsah skládat z jednotlivých služeb a jejich popisu. Rezervace nás zavede k rezervačnímu formuláři, kde návštěvník může provést rezervaci. Fotogalerie nám otevře galerii salónu, různých služeb a podobně. Kontakt poskytuje informace, jak kontaktovat salón. Banner zde slouží pouze jako grafický prvek na přivítání nově příchozích uživatelů.

Obr. 18: Domovská stránka a navigace

Zdroj: Vlastní

3.2.2 Sekce s odkazy na různé služby

Sekce má upoutat zákazníkovu pozornost na jednotlivé služby, které kosmetička nabízí, jejich krátkým, chytlavým popisem. Po kliknutí na zadanou službu na ni bude zákazník přesměrován na stránce „Služby“. Zde se také budou realizovat animace při najetí myší.



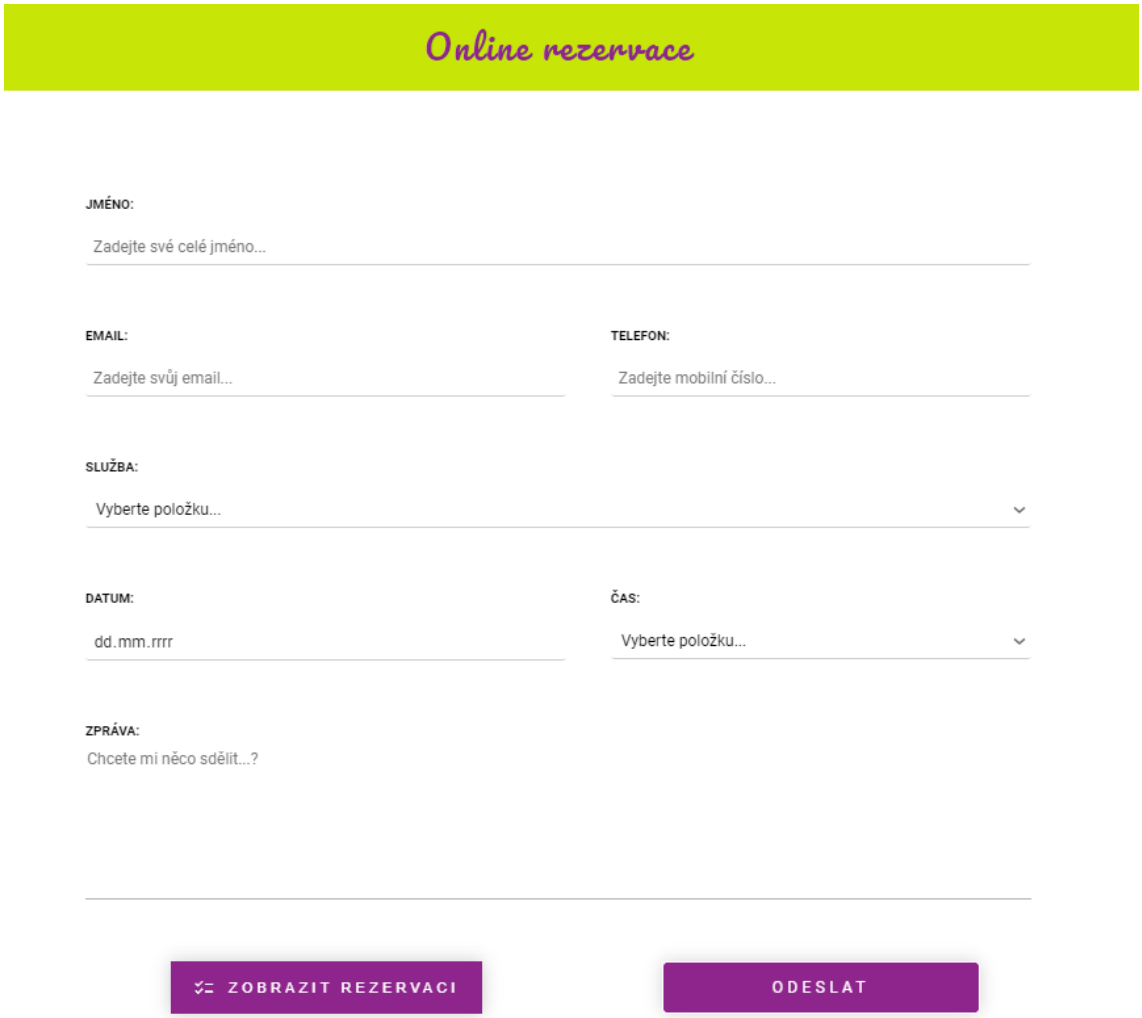
Obr. 19: Sekce služby

Zdroj: Vlastní

3.2.3 Online rezervační formulář

Další sekcí je online formulář, který slouží pro zarezervování data, času a různých služeb, které si návštěvník vybere. Poté bude moc rezervaci odeslat a ta se zapíše do systému kosmetičky. Formulář bude mít i svoji vlastní kontrolu, aby uživatel nemohl zadávat nesmyslné informace.

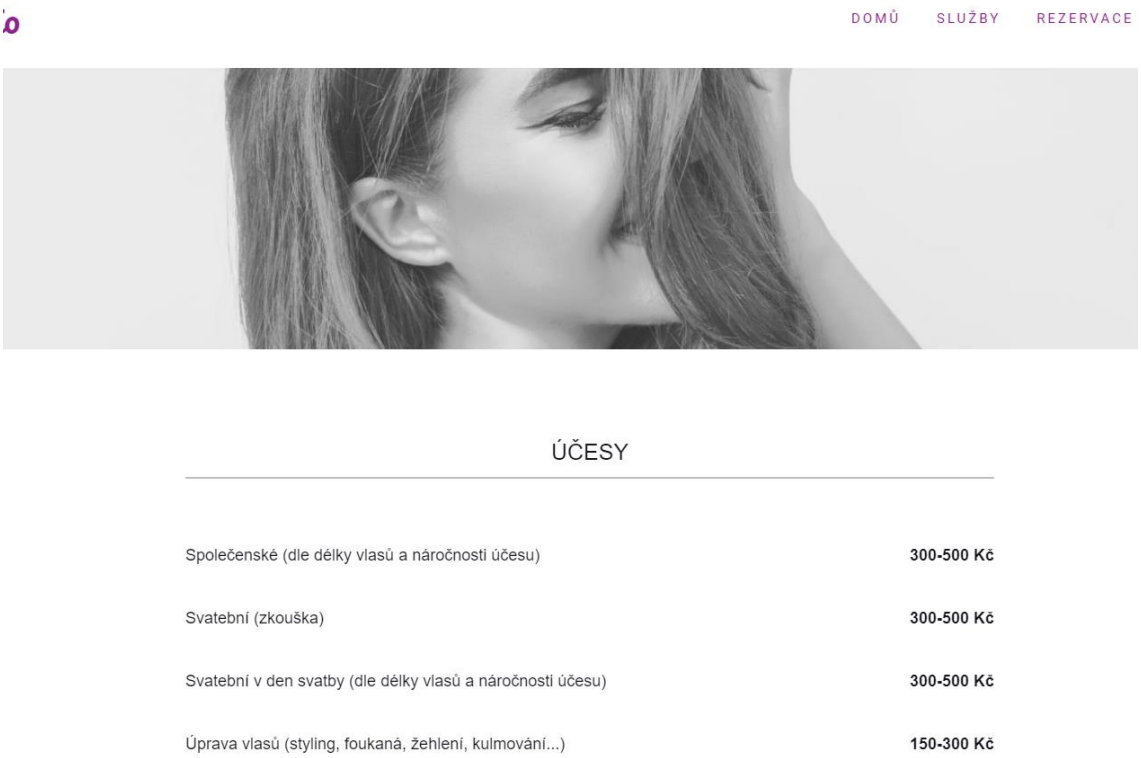
Obr. 20: Rezervační formulář



Zdroj: Vlastní

3.2.4 Stránka služby

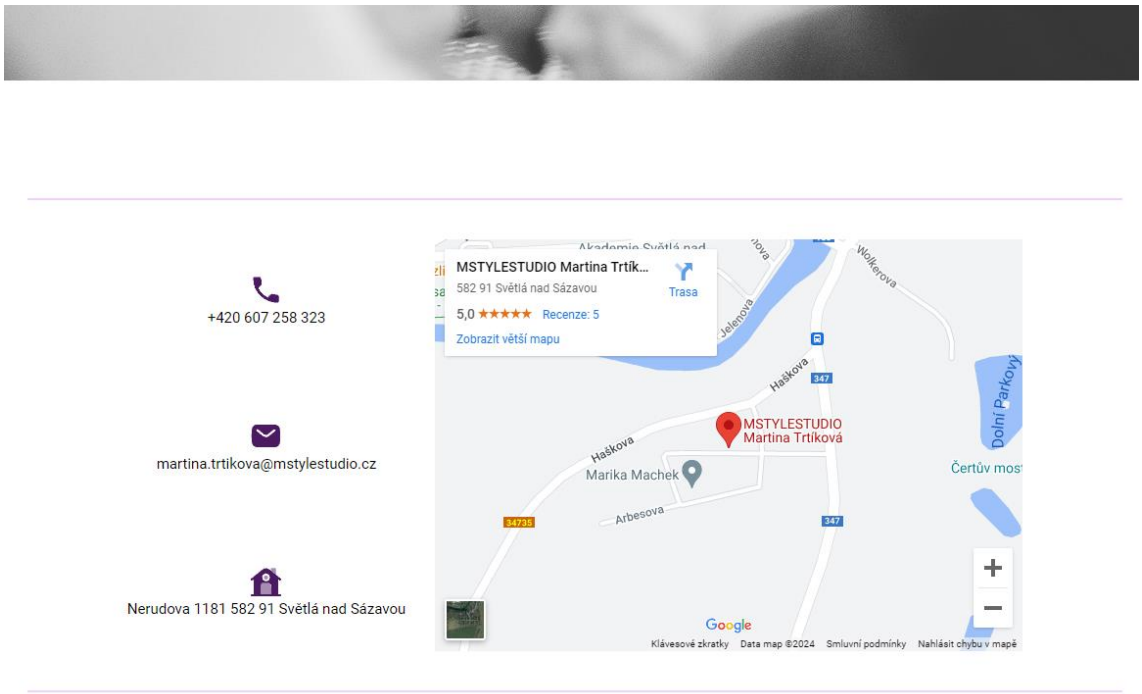
Stránka služby bude obsahovat základní údaje o službách a jejich cenových nabídkách. Tímto designem bude realizována každá služba. Stránka bude obsahovat animace. Tímto stylem designu bude dosaženo jednoduchosti a přehlednosti jednotlivých služeb.



Obr. 21: Stránka služby
Zdroj: Vlastní

3.2.5 Stránka kontakt

V této záložce budou obsahem základní kontaktní informace pro návštěvníky stránek. Stránka také bude obsahovat mapu s přesným místem studia kosmetičky.

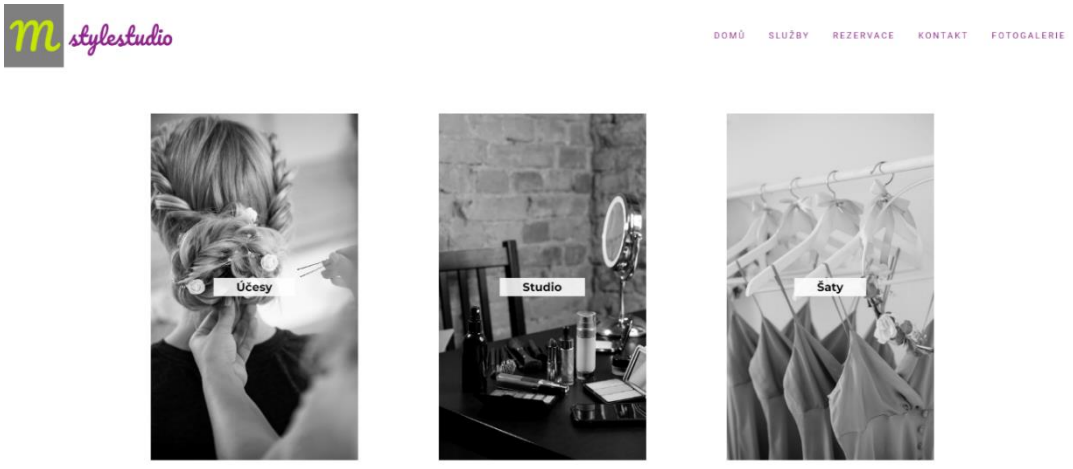


Obr. 22: Stránka kontakt

Zdroj: Vlastní

3.2.6 Stránka fotogalerie

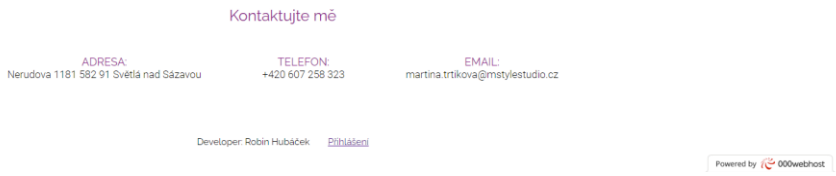
V této sekci najdeme jednoduchý design fotogalerie, kde si návštěvník bude moci kliknout na určitý typ galerie, a tím si zobrazit různé fotografie kosmetičky.



Obr. 23: Stránka fotogalerie

Zdroj: Vlastní

Poslední důležitou komponentou je patička webové stránky, kde se nachází odkaz na přihlášení do systému s rezervacemi. Zde se bude moci kosmetička přihlašovat se svými údaji do systému a spravovat jej.



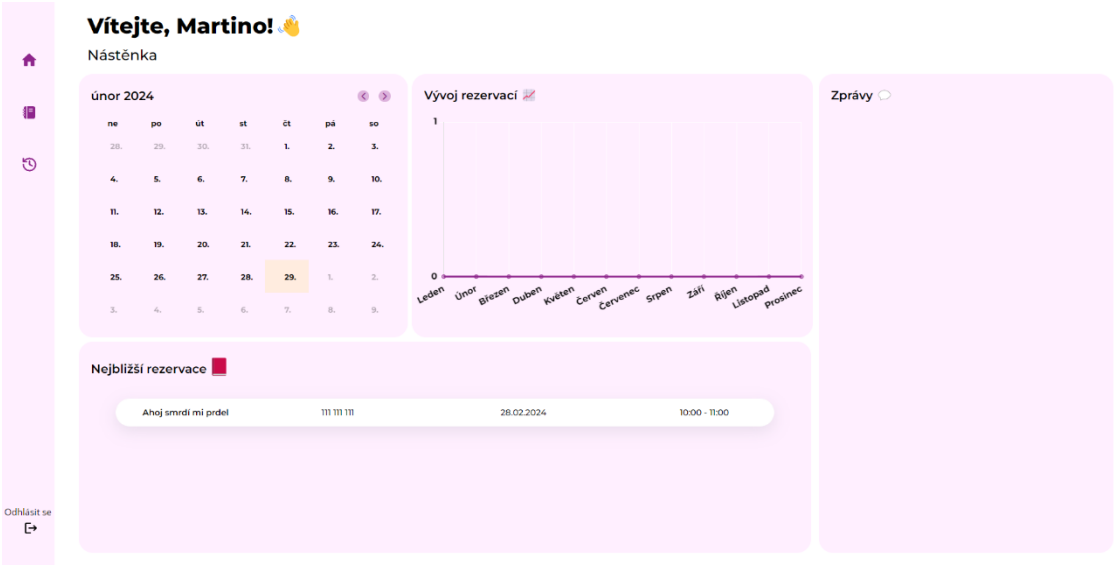
Obr. 24: Patička

Zdroj: Vlastní

Tímto je hotový design webových stránek. Stránky jsou dostupné na testovacím webhostingu na adrese https://mstylestudio.000webhostapp.com.

3.2.7 Design admin panelu

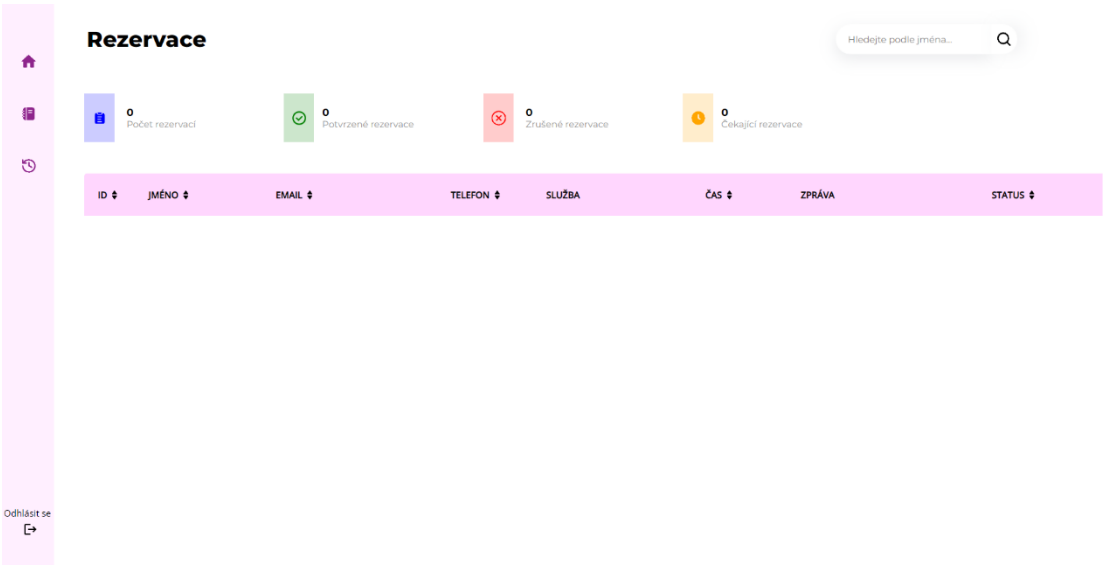
Admin panel byl vytvořen úplně od začátku bez žádných šablon. Obsahuje základní informace, které můžete vidět na obrázku, jsou jimi například kalendář pro zadávání volna pro kosmetičku a dalších informací. Podle toho, které dny má volno, budou vypnuté rezervace. Dále je zde graf pro přehled, kolik rezervací bylo v jednotlivých měsících provedeno. Najdeme zde také zprávy, které zaslali zákazníci, a může na ně být odpovězeno.



Obr. 25: Admin panel

Zdroj: Vlastní

Dalším sektorem v admin panelu jsou rezervace. Zde kosmetička bude moci přepínat statusy jednotlivých rezervací a tím zasílat emaily o potvrzení rezervace nebo její zrušení. Zde bude také moci vyhledávat jednotlivé rezervace nebo je různě filtrovat.



Obr. 26: Rezervace

Zdroj: Vlastní

Posledním sektorem je historie rezervací, ta bude ukazovat všechny rezervace, jež byly doposud odeslány.

3.3 Tvorba webu

V této sekci se budu detailněji zabývat tvořením stránek. Budou předvedeny příklady kódů a různé typy řešení, jak byly jednotlivé stránky vytvořeny. K tomu bude zapotřebí značkovací jazyk HTML, pro design CSS a pro dynamiku stránky JavaScript. Samozřejmě budu používat Webpack a preprocesor SCSS. Webpack využiji pro lepší organizaci a celkovou efektivitu práce pro dosažení stanovených cílů.

3.3.1 Instalace a konfigurace webpacku

V prvním kroku musíme aplikovat Webpack, k tomu využijeme správce balíčků NPM. V terminálu Visual Studio Code napíšeme následující příkazy.

```
npm install --save-dev webpack
npm install --save-dev webpack-cli
```

Těmito příkazy se zajistí, že se stáhne nejnovější verze Webpacku a jeho funkce. Dále si musíme vytvořit konfigurační soubor, který nám specifikuje vstupy, výstupy, pluginy a moduly. Pro vlastní potřeby byl vytvořen specifický konfigurační soubor a z důvodu velkého objemu zde bude ukázka pouze části.

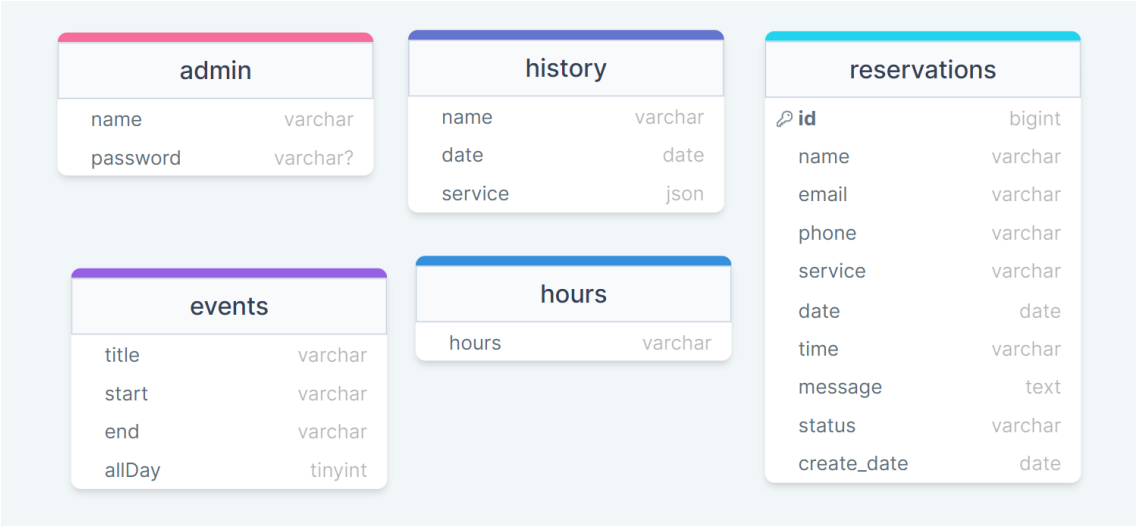
```
const public = {
  mode: 'development',
  entry: {
```

```
    index: ["./src/layouts/home/index.js",
"./src/layouts/home/home.scss"],
    services: ["./src/layouts/services/services.js",
"./src/layouts/services/services.scss"],
    contact: ["./src/layouts/contact/contact.js",
"./src/layouts/contact/contact.scss"],
    photo: ["./src/layouts/photo/photo.js",
"./src/layouts/photo/photo.scss"],
    login: ["./src/layouts/login/login.js",
"./src/layouts/login/login.scss"]
  },
  output: {
    filename: "[name].js",
  },
  module: {
    rules: [
      {
        test: /\.(s[ac]|c)ss$/i,
        use: [
          MiniCssExtractPlugin.loader,
          "css-loader",
          "sass-loader"
        ],
      },
      {
        test: /\.(png|svg|jpg|jpeg|gif)$/i,
        type: 'asset/resource',
      },
      {
        test: /\.(woff|woff2|eot|ttf|otf)$/i,
        type: 'asset/resource',
      },
    ],
  },
```

Zde jednoduše načítáme obrázky, CSS, fonty a HTML soubory. Můj konfigurační soubor dále obsahuje modul pro soubory admina a dále pluginy pro načítání HTML souborů a importování stylů SCSS. Dále si vytvoříme strukturu projektu a tím bude složka „source“ zkráceně „src“ a „distribution“ zkráceně „dist“. V source budou všechny soubory pro realizaci stránky a do složky dist se automaticky generují soubory, které poté už jen jednoduše vložíme na webhosting.

3.3.2 Rezervační systém

Rezervační systém byl realizován prostřednictvím PHP, tedy mého back-end jazyku. Je postaven na dotazech na server s databází. Nejdříve byl vytvořen diagram návrhu tabulek.



Obr. 27: Schéma tabulek

Zdroj: Vlastní

Toto je jednoduchý návrh tabulek, abych si upřesnil, která data jakého typu chci uchovávat.

Tabulka admin nám bude uchovávat přihlašovací informace do admin panelu, který bude využívat pouze kosmetička. Vytvořil jsem pro ni přihlašovací jméno a také heslo, které jsem zahashoval. Zahashování je proces, při kterém se původní data (jako heslo) transformují do jiného formátu pomocí matematické funkce nazývané hašovací funkce. Výsledkem je nečitelný řetězec znaků, nazývaný hash, který není možné jednoduše převést zpět na původní data. Hashování jsem provedl pomocí jazyka PHP, který má v sobě zabudovanou funkci hash(). Zde je ukázka funkce hash().

```php
<?php
echo hash('sha256', 'The quick brown fox jumped over the lazy dog.');
?>
```

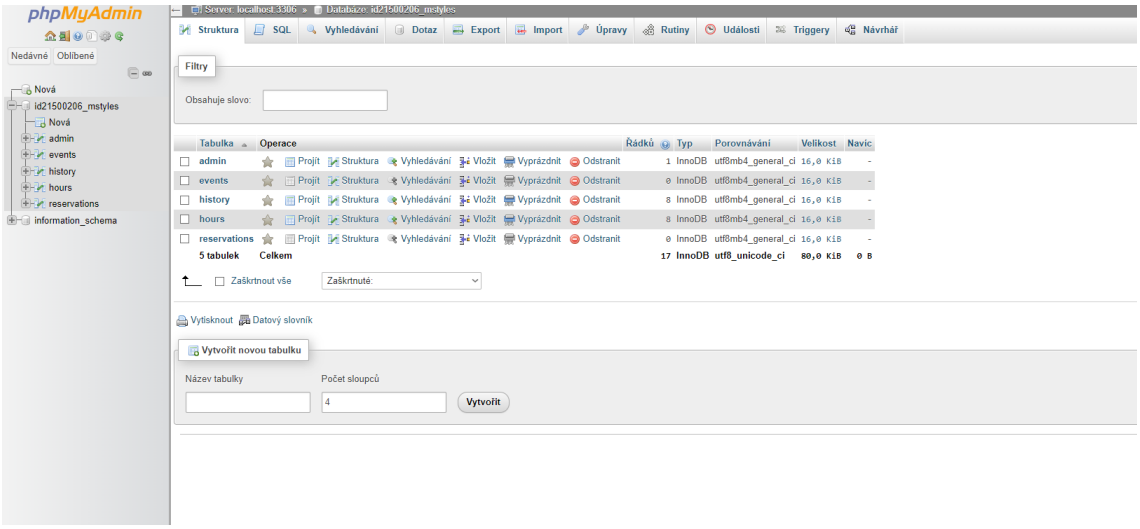
Výsledek: 68b1282b91de2c054c36629cb8dd447f12f096d3e3c587978dc2248444633483 (Zahashovaný text)

Obr. 28: Ukázka hash funkce

Zdroj: Vlastní

Další tabulkou je reservations. Do této tabulky se budou přidávat jednotlivé rezervace od zákazníků. Uchovává se zde ID (Identifikační číslo) rezervace, dále jméno, e-mail, mobil, služby, datum, čas, zpráva, status rezervace a nakonec, kdy byla rezervace vytvořena. V events se ukládají data z admin kalendáře, kde si kosmetička vybírá, která data budou pro rezervace vypnuta, a nebude tedy možné v nich rezervace provádět. Hours slouží k ukládání časů, v nichž kosmetička pracuje. Tím se zajistí, že si návštěvník bude moci rezervovat specifický čas ve specifickém datu. Tabulka history slouží pouze jako tabulka pro uchovávání všech doposud zaregistrovaných rezervací.

Tento následný návrh jsem implementoval do PhpMyAdmin, který je dostupný na mém webhostingu. S touto databází bude poté komunikovat můj front-end.



Obr. 29: Ukázka PhpMyAdmin

Zdroj: Vlastní

3.3.3 Tvoření obsahu stránek

Stránky byly vytvořeny pomocí komponent. Komponenty webových stránek jsou jednotlivé prvky, které tvoří strukturu a obsah webových stránek. Tyto komponenty jsou navrhovány a implementovány tak, aby poskytovaly uživatelům interaktivní a přehledné prostředí při prohlížení webové stránky. Byly realizovány pomocí JavaScriptu, abych je mohl opakovaně používat. Příkladem může být komponenta navigace, která je jako samostatný JavaScript soubor.

```
//Import dat - menu
import { menu_data } from "../../data/home-data/navData";
import { logo } from "../../data/home-data/navData";
import { menuAnchors } from "../../data/home-data/navData";

export function Navbar() {
    //Generování a inicializování proměnných
    const logo_container = document.createElement('div');
    const header = document.createElement('header');
    const nav = document.createElement('nav');
    const nav_list = document.createElement('ul');
    const logo_label = document.createElement('h1');

    //Vytvoření hamburger ikony
    const toggleIcon = document.createElement('button');

    //Třídy pro jednotlivé proměnné
    logo_container.classList.add('logo-container', 'flex-center-row');
    logo_label.classList.add('logo-label', 'flex-center-row');

    toggleIcon.classList.add('nav-icon');
    toggleIcon.setAttribute('aria-expanded', false);

    nav.setAttribute('data-visible', false);
```

```
    //Vkládání obsahu do proměnných (text...apod)
    logo_container.innerText = logo.letter;
    logo_label.innerText = logo.label;

    header.classList.add('primary-navigation', 'flex-center-row');
    logo_label.classList.add('text-primary-clr', 'text-pcf');
    logo_container.classList.add('text-pcf', 'text-second-clr');

    //Generování obsahu navigace | menu
    menu_data.map((name, index) => {
        const nav_item = document.createElement('li');
        const anchor = document.createElement('a');

        nav_item.classList.add('nav-item', 'text-center');
        nav_list.classList.add('nav-list', 'flex-center-row');
        nav.classList.add('nav-bar');

        anchor.innerText = name;
        anchor.href = menuAnchors[index];
        anchor.classList.add('text-rob', 'text-primary-clr', 'text-s-
150', 'text-upper');

        nav_item.appendChild(anchor);
        nav_list.appendChild(nav_item);
    });
    nav.appendChild(nav_list);
    header.appendChild(logo_container);
    header.appendChild(logo_label);
    header.appendChild(toggleIcon);
    header.appendChild(nav);
    return header;}
```

Kód vytvoří komponentu pro navigaci, tu mohu dále použít na ostatních stránkách, a nemusím ji tak vytvářet znovu. Tento způsob tvoření komponent mi usnadnil práci a ušetřil čas. Takto byly vytvořeny i ostatní prvky na stránkách a v panelu Admina.

Import a export jsou zavedené syntaxe pro modulární programování v JavaScriptu. Tyto konstrukce umožňují importovat a exportovat funkce, objekty, třídy a další prvky mezi různými moduly. Dříve byla JavaScriptová aplikace často napsána jako jeden velký soubor kódu, což mohlo být obtížné udržovat a škálovat. S použitím importu a exportu mohu rozdělit kód do menších, snadno spravovatelných částí, což výrazně usnadňuje práci na rozsáhlejších projektech. Komponentu exportuji tím, že před funkci napíšu klíčové slovo „export“. Dále naimportuji v dalším JavaScriptovém souboru, kde chci komponentu navigace použít.

```
import { Navbar } from '/src/components/navigation/navbar';
```

Dále už stačí funkci zavolat a na stránce se nám vykreslí navigace.

```
document.body.insertBefore(Navbar(), content);
```

3.3.4 Stylizování

Vzhled jednotlivých komponent a prvků na stránkách byl realizován pomocí preprecesoru SCSS, a pokud se styly měnily dynamicky, bylo zapotřebí programovacího jazyku JavaScript.

Nejzajímavější věcí je animace při setkání uživatele s komponenty stránky. Tento způsob animace používá mnoho webů a řadí se mezi moderní prvky novodobého webového vývoje. Animace byla realizována pomocí JavaScriptu a malého bloku SCSS.

```
function intersectObserver() {
      const observer = new IntersectionObserver((entries) => {
        entries.forEach((entry) => {
          if (entry.isIntersecting) {
            entry.target.classList.add("show");
          }
        });
      });
      const hiddenElements = document.querySelectorAll(".hidden");
      hiddenElements.forEach((el) => observer.observe(el));
}
```

Tento kód implementuje funkci IntersectObserver, která využívá Intersection Observer API, což je rozhraní v prohlížeči umožňující sledovat, když se elementy kříží s daným kontejnerem nebo viewportem prohlížeče. Funkce začíná vytvořením instance IntersectionObserver, která přijímá funkci zpětného volání. Tato funkce je spuštěna pro každý vstup (entry), který je sledován.

V těle této funkce se iteruje přes všechny vstupy (entries), které jsou vráceny IntersectionObserver. Pro každý vstup se kontroluje, zda je aktuálně v průniku s view-portem (uživatelským pohledem) pomocí podmínky if (entry.isIntersecting). Pokud je to pravda, přidá se cílovému prvku (target) třída "show". Tímto způsobem lze provádět různé akce nebo změny stylů na prvcích, když jsou viditelné v prohlížeči. Nakonec se funkce intersectObserver volá pro všechny prvky s třídou "hidden", které mají být sledovány a měněny, když jsou viditelné.

Dále už jen stačilo definovat SCSS pro třídy hidden a show.

```
.hidden {
    opacity: 0;
    transition: all 1s;
    filter: blur(5px);
    transform: translateX(-100%);
}
.show {
    opacity: 1;
    filter: blur(0);
    transform: translateX(0%);
}
```

Třída hidden v CSS definuje styly pro prvky, které jsou implicitně skryté na webové stránce. Název třídy "hidden" naznačuje, že tyto prvky nejsou viditelné při načítání stránky, ale mohou být aktivovány později pomocí interakce uživatele nebo jiných událostí. V tomto případě to je pomocí intersektování v předchozím kódu.

Styly definované pro třídu .hidden zahrnují nastavení průhlednosti na nulu pomocí opacity: 0;, což způsobí, že prvky nejsou viditelné. Dále je zde definován přechodový efekt pro změny vlastností prvků s třídou .hidden pomocí transition: all 1s;, který animuje veškeré změny

s trváním jedné sekundy. Dále je aplikováno rozostření na prvky pomocí filter: blur(5px); a posunutí prvků mimo obrazovku vlevo pomocí transform: translateX(-100%);.

Třída show je opakem třídy hidden. Prvky dá do původního stavu a tímto způsobem je realizována animace prvků.

3.3.5 Rezervační formulář

Rezervační formulář jako komponenta je nejdůležitější prvek na stránkách a byla také nejobtížnější na vytvoření. Musel jsem zde dbát na to, aby byl co nejjednodušší z pohledu designu a funkčnosti. Prvním krokem bylo navrhnout strukturu, jak bude formulář vypadat. Dalším důležitým bodem byla jeho kontrola, kterou jsem realizoval pomocí JavaScriptu. Ukázka kódu pro validaci formulářových prvků:

```
const isValidPhone = (phone) => {
      const re = /^\d{3} \d{3} \d{3}$/;
      return re.test(phone);
};
```

Toto je jeden z mnoha kódů, který slouží k validaci, zda je mobilní číslo ve správném číselném tvaru, či nikoliv. Podobné typy kódů jsou vytvořeny i pro zbylé formulářové prvky jako je email, jméno a ostatní.

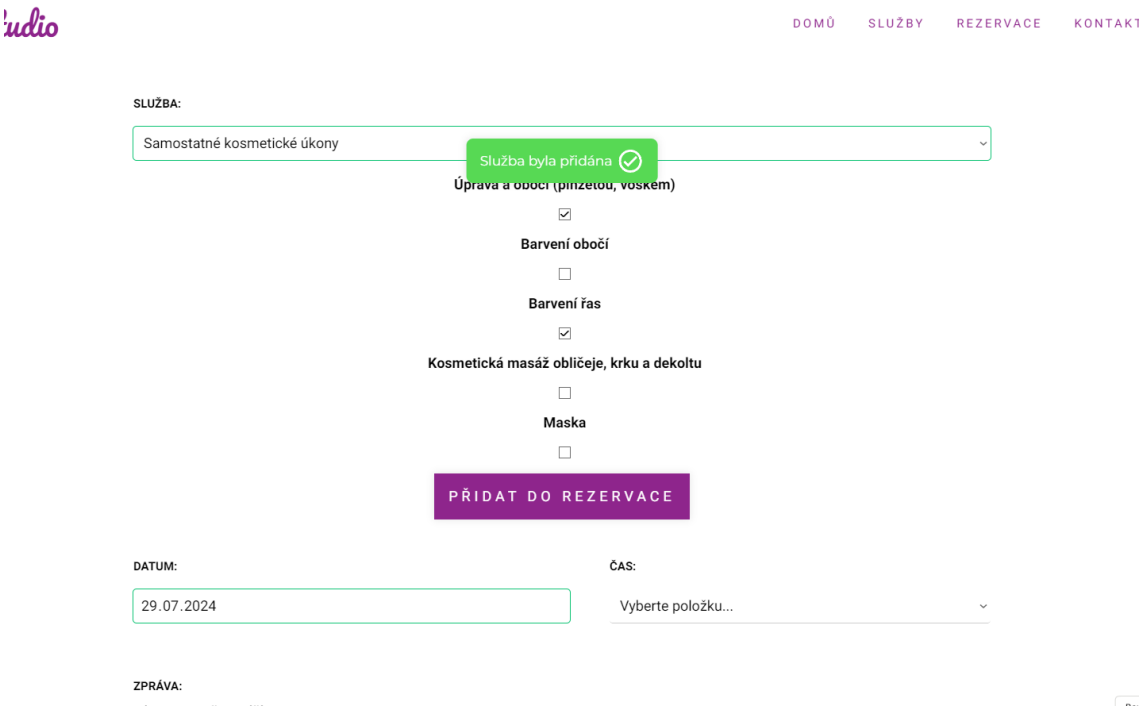
Využívám regulární výrazy při validaci uživatelského vstupu na rezervačním formuláři. Když uživatel vyplňuje informace jako jméno, e-mailová adresa, telefonní číslo nebo datum, chci zajistit, aby tyto údaje byly ve správném formátu, aby bylo možné rezervaci úspěšně zpracovat.

Například, pomocí regulárního výrazu mohu ověřit, zda je e-mailová adresa zadaná ve formátu "example@example.com". Tímto způsobem se vyhneme chybám při zadávání e-mailů.

Podobně mohu také použít regulární výrazy k ověření správnosti telefonních čísel, aby odpovídala určitému formátu, jako je například český formát "+420123456789". To zajišťuje, že můžeme snadno kontaktovat uživatele v případě potřeby.

Regulární výrazy jsou tedy důležitou součástí rezervačního formuláře, která pomáhá zabezpečit, že uživatelé poskytují platné a správně formátované údaje, což zlepšuje celkovou spolehlivost a uživatelskou zkušenost s formulářem.

Následně jsem musel vytvořit tlačítko pro přidávání jednotlivých služeb. Toto tlačítko je nastaveno tak, že se objeví pouze, pokud uživatel vybere určitou službu a dále se mu objeví další specifikace o službě.



Obr. 30: Notifikace

Zdroj: Vlastní

Pokud uživatel klikne na tlačítko přidání do rezervace, tak se mu objeví úspěšná zpráva o její přidání do rezervace. Pokud ale tato služba již v rezervaci je, tak se mu objeví chybová zpráva o tom, že tato služba je již zarezervována.

Tlačítko zobrazit rezervaci ukáže uživateli veškeré služby, které si vybral. Jednotlivé služby může mazat.

Poslední tlačítko, které se zde nachází, je tlačítko pro odeslání. Pokud na něj uživatel klikne, odešle se požadavek na server. Tento požadavek s sebou nese informace o rezervaci, která byla zaslána a následně se ji snaží přidat do systému pro rezervace. Pokud byly informace v pořádku a na serveru nejsou problémy, uživatel obdrží úspěšnou hlášku, že rezervace byla odeslána.

3.3.6 Kalendář

Dalším problémem bylo najít způsob, jakým bude kosmetička vypínat rezervace, pokud to bude potřeba. Využil jsem k tomu kalendář, který jsem nainstaloval pomocí správce balíčků NPM. Jelikož jich ve správci existuje mnoho, musel jsem vybrat balíček vyhovující mým kritériím. Použil jsem „FullCalendar“, ten jsem poté musel nainstalovat pomocí správce příkazem:

```
npm i fullcalendar
```

Ke každému balíčku je napsána i dokumentace a návody, jak s těmito komponentami pracovat. Využil jsem oficiální dokumentace pro jeho vytvoření a konfiguraci.

Obr. 31: Kalendář

Zdroj: Vlastní

Konfigurace zahrnovala výběr barvy, typ písma, přepínání jednotlivých měsíců, jaký jazyk se má zobrazovat a více. Malá ukázka kódu konfigurace:

```
const calendar = new Calendar(calendarEl, {
      timeZone: "Europe/Prague",
      plugins: [interactionPlugin],
      initialView: "dayGridMonth",
      aspectRatio: 1,
      themeSystem: "bootstrap5",
      locale: "cs",
      selectable: true,
      headerToolbar: {
        start: "title",
        end: "prev,next",
      },
      buttonText: {
        today: "Dnes",
        month: "Měsíc",
        week: "Týden",
        year: "Rok",
        day: "Den",
      },
```

V dalším kroku jsem musel nakonfigurovat, co se má stát, pokud kosmetička vybere jedno datum klikem, či vybere více dní potažením. Po libovolném označení vyskočí okno, které očekává název události pro vybraná data. Po vyplnění názvu a potvrzení, se v kalendáři objeví od kdy do kdy daná událost trvá. Data se poté uloží do databáze. Pokud návštěvník poté vybere jedno z těchto dat, zobrazí se mu zpráva, že rezervace jsou pro tato data vypnuty.

Graf, který je na Admin panelu, byl vytvořen stejným způsobem. Nejprve jsme museli přes NPM správce balíčků nainstalovat libovolný nástroj, který se zabývá grafy. Dále už stačila pouze jeho konfigurace, a to je například jazyk, osa y, osa x. Graf má ukazovat vývoj rezervací v jednotlivých měsících, tato data získávám prostřednictvím PHP skriptu.

3.3.7 Carousel

Na stránce fotogalerie jsem využil komponenty, takzvané carousel. Carousel je interaktivní prvek užívaný k prezentaci obsahu, jako jsou obrázky, videa nebo jiné mediální prvky. Typicky se jedná o řadu snímků, které se posouvají horizontálně nebo vertikálně, obvykle s možností ručního nebo automatického přepínání mezi jednotlivými snímky.

Nachází se zde tři prvky, na něž může uživatel kliknout – účesy, studio a šaty. Po kliknutí se uživateli zobrazí sada obrázků podle toho, kterou kategorii zvolil. Následně se pomocí Fetch API odešle dotaz na server neboli na PHP skript.

```
fetch(`./database/loadImages.php?gallery=${galleryType}`, {
      method: "GET",
})
```

Zde jsem využil metodu GET, protože chci, aby mi server poskytl data. Tato data jsou specifikována typem galerie, na níž uživatel klikl. Každý ze tří prvků obsahuje atribut data-images, do kterého je zapsáno, o jakou galerii se jedná. Podle tohoto atributu skript vybere složku pro typ galerie a následně vybere obrázky, které mi prostřednictvím dotazu zašle.

```
const ImageSlider = {
  RenderImage: (data, gallery) => {
    const image = document.querySelector(".image");
    const url = `./images/${gallery}/${data[current]}`;
    image.src = url;
  },
};
```

Tento kód naplní HTML tag <img> atributem src, který obsahuje odkaz na obrázek z galerie. Následně uživatel může manuálně přepínat obrázky, které obsahuje daná galerie.

3.4 Implementace back-endu

Jelikož je můj rezervační systém postaven na dotazech na moji serverovou databázi, musel jsem vytvořit PHP skripty, skrze něž můj front-end komunikuje. Následující kódy jsou ukázkami mnoha skriptů, jež moje stránky využívají.

3.4.1 Odesílání rezervací

```
<?php
$servername = "localhost";
$username = "admin_mstyle";
$password = "MstyleAdmin";
$database = "mstyles";
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    http_response_code(400);
```

```
    echo json_encode(['error' => 'Neplatná metoda požadavku']);
    exit;
}
$conn = new mysqli($servername, $username, $password, $database);
$jsonData = file_get_contents('php://input');
$data = json_decode($jsonData, true);
if ($data === null) {
    http_response_code(400);
    echo json_encode(['error' => 'Chybný formát dat']);
    exit;
}
$name = $data['Jméno'];
$email = $data['Email'];
$phone = $data['Telefon'];
$date = $data['Datum'];
$time = $data['Čas'];
$message = $data['Zpráva'];
$servicesArr = json_encode($data['Služby'], JSON_UNESCAPED_UNICODE);
$sqlReservations = "INSERT INTO reservations (name, email, phone,
date, time, message, service, create_date)
                    VALUES ('$name', '$email', '$phone', '$date',
'$time', '$message', '$servicesArr', NOW())";
if ($conn->connect_error) {
    http_response_code(500);
    echo json_encode(['error' => 'Chyba připojení k databázi: ' .
$conn->connect_error]);
    exit;
}
if ($conn->query($sqlReservations) === TRUE) {
    $response = ['status' => 'success', 'message' => 'Rezervace byla
úspěšně odeslána.'];
    echo json_encode($response);
} else {
    http_response_code(500);
    echo json_encode(['error' => 'Chyba při vkládání záznamu do
databáze: ' . $conn->error]);
```

Tento PHP skript slouží k zpracování dat z formuláře, který je odeslán na server. Nejprve se nastavují informace pro připojení k databázi, jako je název serveru, uživatelské jméno, heslo a název databáze. Poté skript ověřuje, zda byl formulář odeslán metodou POST. Pokud nebyl, vrátí chybovou zprávu a HTTP odpověď s kódem 400.

Pokud byl formulář odeslán metodou POST, skript se pokusí připojit k databázi pomocí zadaných údajů. Poté zpracuje data z formuláře, která jsou přijata jako JSON. Pokud jsou data ve správném formátu, uloží je do proměnných. Tyto proměnné obsahují informace jako je jméno, e-mail, telefonní číslo, datum, čas, zpráva a seznam služeb.

Následně se sestaví SQL dotaz pro vložení rezervace do databáze. Tento dotaz obsahuje informace o rezervaci, jako jsou jméno, e-mail, telefonní číslo, datum, čas, zpráva a seznam

služeb. Dotaz je proveden a v případě úspěchu je vrácena odpověď ve formátu JSON s informací o úspěšném odeslání rezervace. V opačném případě je vrácena chybová zpráva a HTTP odpověď s kódem 500, která oznamuje problém s vkládáním záznamu do databáze.

Dalším důležitým úkolem bylo zajistit, aby si uživatel nemohl vybrat stejný čas a datum, které už v databázi jsou, tím by vznikaly kolize v rozvrhu kosmetičky. Následující skript zajišťuje, aby se tento případ nemohl stát.

3.4.2 Volné a zabrané časy

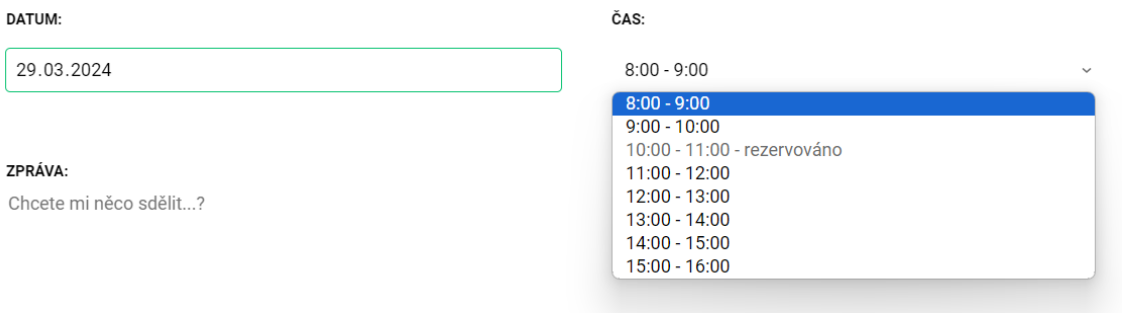
```
<?php
$servername = "localhost";
$usernameDB = "admin_mstyle";
$passwordDB = "MstyleAdmin";
$database = "mstyles";

$jsonData = file_get_contents('php://input');
$requestData = json_decode($jsonData, true);
$date = $requestData['date'];
$conn = new mysqli($servername, $usernameDB, $passwordDB, $database);
if ($conn->connect_error) {
    die("Connection to database failed: " . $conn->connect_error);
}
$sql = "SELECT h.hours,
        CASE WHEN r.time IS NULL THEN 'volný' ELSE 'zabraný' END AS
status
        FROM hours h
        LEFT JOIN reservations r ON h.hours = r.time AND r.date =
'$date'
        ORDER BY STR_TO_DATE(h.hours, '%H:%i')";
$result = $conn->query($sql);
$responseData = array();
if ($result) {
    while ($row = $result->fetch_assoc()) {
        $data[] = array('hours' => $row['hours'], 'status' =>
$row['status']);
    }
    echo json_encode($data);
} else {
    echo "Error executing query: " . $conn->error;
}
$conn->close();
?>
```

PHP skript zpracovává data ve formátu JSON, která jsou odeslána pomocí HTTP POST metody. Nejprve načte tato data a dekóduje je do PHP pole. Následně extrahuje datum z těchto dat a přistupuje k databázi pomocí objektu mysqli, který reprezentuje spojení s MySQL databází.

Pokud se připojení k databázi nezdaří, skript ukončí běh a vypíše chybu. Pokud je připojení úspěšné, sestaví SQL dotaz pro získání informací o hodinách a jejich stavu pro dané datum.

Tento dotaz kombinuje data z tabulek hours a reservations, abychom zjistili, zda jsou určité hodiny zabrané nebo volné. Výsledek tohoto dotazu je následně zpracován a vrácen ve formátu JSON s informacemi o stavu každé hodiny pro dané datum. Výstup je poté spojen s front-endem a uživateli se ukáže, se ukáže, které hodiny jsou ve vybraném datu volné.



Obr. 31: Rezervační systém – výběr času

Zdroj: Vlastní

Pokud při zpracování dotazu nastane chyba, skript vypíše chybovou zprávu. Nakonec se uzavře spojení s databází a skript ukončí svůj běh.

3.4.3 Stav rezervací

Následující řešení je spojeno s problémem, aby kosmetička mohla rezervace potvrdit a zrušit. To je řešeno v sekci Rezervace v Admin panelu, kde sloupeček „status“ ukazuje, které rezervace čekají na potvrzení, nebo zrušení. Kosmetičce se poté zobrazí okno, kde může napsat důvod zrušení rezervace nebo doplňující informace k potvrzení rezervace. Po vyplnění a stisknutí tlačítka pro odeslání se v databázi přepíše status rezervace a dále se odešlou data do PHP skriptu, který se stará o odesílání e-mailu klientovi.

4 Srovnání s existujícím řešením

Webové stránky, jež vizážistické studio nyní využívá, neobsahuje rezervační systém, což je pro kosmetičku namáhavější z hlediska její pracovní organizace. Mým hlavním cílem tedy bylo vytvořit a navrhnout systém, který jí ulehčí tento problém a ušetří čas.

Jedním z dalších úkolů byla modernizace a celkové zlepšení fungování webové stránky. V této digitální době je klíčové udržet krok s neustále se vyvíjejícími technologiemi a očekáváními uživatelů. Proto bylo nutné provést revizi stávajícího webu s cílem zvýšit jeho efektivitu, responzivitu a uživatelskou přívětivost. Tato modernizace zahrnovala aktualizaci designu, optimalizaci rychlosti načítání stránek, zavedení nových funkcí pro zlepšení interaktivity a zajištění kompatibility s různými zařízeními a prohlížeči. Tyto úpravy byly provedeny s cílem zajistit, že web nejenom vizuálně osloví uživatele, ale také poskytne pohodlný a efektivní prostředek pro dosažení jejich cílů a potřeb online.

Pro zpracování tohoto projektu jsem zvolil metodu Waterfall (vodopádový model) pro řízení projektů, který popisuje vývojový proces projektu a je rozdělen na jednotlivé fáze.

1. Plánování

Seznámil jsem se s požadavky klienta a stanovil si cíle projektu, kterými bylo zvýšení online přítomnosti, zvýšení rezervací, prezentace jednotlivých služeb, vytvoření rezervačního systému a další. Dále jsem musel identifikovat cílovou skupinu, abych vytvořil vhodné stránky a správně optimalizované stránky.

2. Analýza

Sestavil jsem si seznam funkcí, jež by měl web obsahovat, například seznam nabízených služeb, fotogalerii, rezervační formulář, kontaktní informace. Dále jsem sestavil obsah a budoucí strukturu stránky. V této fázi byly zvoleny i technologie, které bych měl využít.

3. Návrh

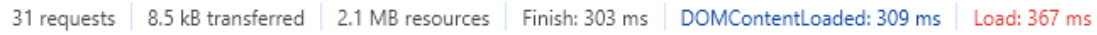
Pomocí grafického softwaru jsem vytvořil návrh stránek. Využil jsem zde jednoduchost a moderní styly pro webové prvky. Musel jsem také navrhnout a promyslet, jaké animace, notifikace bude stránka obsahovat. Jediným specifickým požadavkem, který jsem dostal, bylo udržovat určenou paletu barev.

4. Implementace

Na základě mého návrhu jsem začal s vývojem webu, kde jsem využil zvolené technologie, jako bylo HTML, CSS a JavaScript a ostatní. Poté jsem pokračoval ve vývoji responzivního designu, který by měl dobře fungovat na různých zařízeních, včetně mobilních telefonů a tabletu.

4.1 Testování a vyhodnocení

Aplikace se testovala během jejího vytváření. Využíval jsem vývojářské nástroje, které jsou součástí každého webového prohlížeče. Prohlížeče, na nichž byla aplikace testována, jsou Mozilla Firefox, Google Chrome, OperaGX, Opera. V nástrojích jsem také testoval responzivitu na různých zařízeních a způsob, jak se na nich stránky chovají. Testovaly se různé modely mobilních telefonů a tabletů. Testování obsahovalo také rychlost načítání obrázků, skriptů, kódů a komponent. Snažil jsem se, aby se stránky načítaly pod 3 sekundy na dobrém připojení k internetu, tento úkol se mi podařilo splnit. Po inicializaci webu, který původně trpěl pomalým načítáním, došlo k výraznému zlepšení. Díky optimalizaci kódu a kompresi obrázků se stránka nyní načítá v čase 309 milisekund. Tento dramatický pokrok v rychlosti načítání zaručuje, že uživatelé, kteří byli dříve konfrontováni s pomalým načítáním, teď mohou okamžitě získat přístup k obsahu a užít si vylepšenou uživatelskou zkušenost.

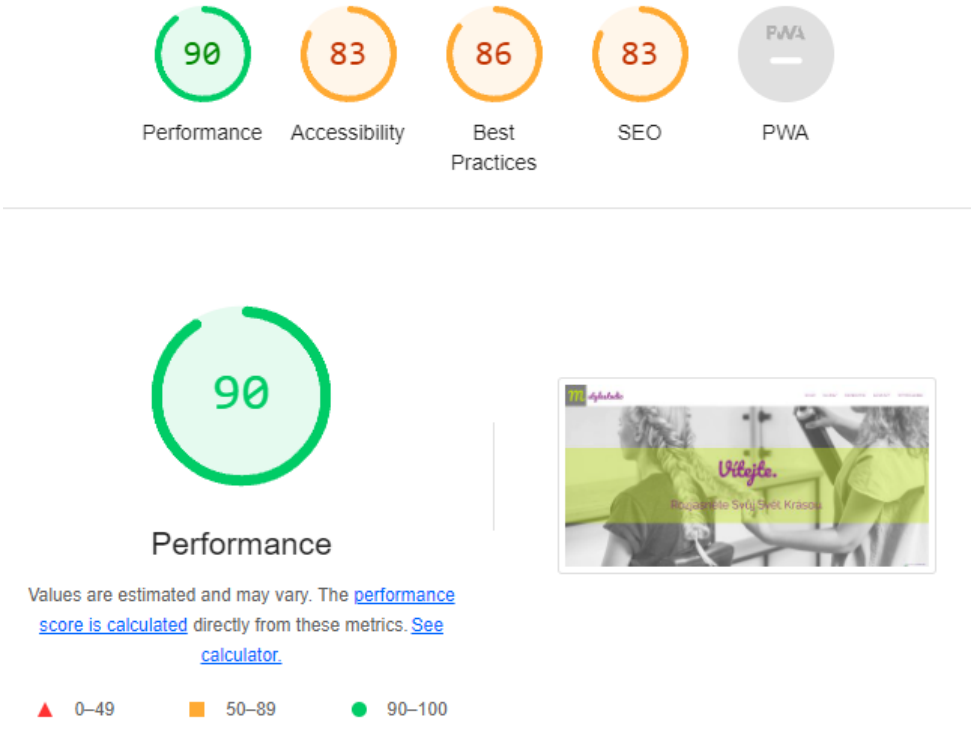


Obr. 32: Načítání stránek

Zdroj: Vlastní

Aplikace bude uložena na webovém hostingu, konkrétně na serveru s určenou doménou. Budu využívat serverovou databázi, kam budu ukládat data.

Nedílnou součástí práce bylo také otestovat stránky pomocí Lighthouse rozšíření do prohlížečů. Lighthouse je automatizovaný nástroj s otevřeným zdrojovým kódem pro zlepšení výkonu, kvality a správnosti webových aplikací. Při auditu stránky provede Lighthouse na stránce řadu testů a poté vygeneruje zprávu o tom, jak dobře si stránka vedla. Odtud můžete použít neúspěšné testy jako ukazatele toho, co můžete udělat pro zlepšení své aplikace.



Obr. 33: Ukázka Lighthouse

Zdroj: Vlastní

Tento výsledek byl testován na testovacím serveru, zjistili jsme však, že existují určité nedostatky v oblasti přístupnosti. Přístupnost ve webovém designu a vývoji se zabývá tím, jak mohou být webové stránky, aplikace a další digitální produkty použitelné pro všechny uživatele, včetně těch s různými typy handicapů, jako jsou zraková, sluchová, motorická nebo kognitivní omezení. Hlavním cílem přístupnosti je zajistit, aby digitální technologie byly snadno použitelné a srozumitelné pro všechny uživatele bez ohledu na jejich schopnosti nebo omezení. Je třeba provést další úpravy, aby byla zajištěna plná přístupnost a inkluzivita pro všechny uživatele. Ostatní nedokonalosti jako je SEO, Best practices, jsou už jen malé nedostatky, které se opraví až při plném spuštění webových stránek. Rezervační systém se dá rozšiřovat, avšak záleží na podnikatelce, co vše bude nyní, nebo v budoucnu potřebovat.

Pevně věřím, že se stránky podaří správně spustit a budou plně funkční. Očekávám, že se bude muset několik maličkostí upravit, aby systém mohl být perfektní.

Závěr

V závěru bakalářské práce bych rád konstatoval, že jsem úspěšně dosáhl všech stanovených cílů. Práce byla zaměřena na vytvoření moderní a funkční webové stránky s rezervačním systémem pro osobu podnikající v oblasti kosmetiky a vlasové péče. Pro dosažení těchto cílů byl proveden důkladný průzkum současného stavu webového prostoru v této oblasti, což posloužilo jako základ pro návrh a implementaci nového designu a funkcí.

Během procesu návrhu a vývoje nového webu jsem dbal na to, aby byly dodrženy základní principy tvorby webových stránek. Kladený důraz na responzivní design, optimalizaci pro různá zařízení a uživatelsky přívětivé rozhraní přispěl k vytvoření stránky, která nejen vizuálně oslovuje, ale také poskytuje pohodlnou a intuitivní interakci pro uživatele.

Nesmírně důležitými nástroji pro realizaci tohoto projektu byly Webpack a npm, jelikož umožnily efektivní správu balíčků a optimalizaci kódu a ulehčily mi tak moji práci a její organizaci.

Záměrem také bylo si osvojit a upevnit znalosti programovacího jazyka JavaScript za účelem budoucí práce v prostředí framework React, ze kterého programovací jazyk vychází, a posunout tak své znalosti o moderním webovém vývoji.

Testování nového webu a vyhodnocení jeho úspěšnosti byly klíčovými kroky při zajištění jeho efektivity. Výsledkem je funkční a moderní webová stránka, která nyní slouží jako účinný nástroj pro propagaci a prodej služeb v oblasti kosmetiky a vlasové péče. Věřím, že se díky této celkové aktualizaci webu zvýší návštěvnost a spokojenost uživatelů.

Jsem rád, že jsem mohl přispět k vytvoření webové stránky, která bude pro klientku efektivním nástrojem pro rozvoj a propagaci jejího podnikání v oblasti kosmetiky a vlasové péče.

Seznam použité literatury

ARMSTRONG, Martin. *Chart: How Many Websites Are There? | Statista*. Online. Statista – The Statistics Portal for Market Data, Market Research and Market Studies. 2021. Dostupné z: https://www.statista.com/chart/19058/number-of-websites-online/. [cit. 2023-12-21].

COURSERA INC. *What Does a Back-End Developer Do? | Coursera*. Online. COURSERA INC. Coursera | Degrees, Certificates, & Free Online Courses. Last modified 29 November 2023. Dostupné z: https://www.coursera.org/articles/back-end-developer. [cit. 2023-12-23].

DAVIS, Michele a PHILLIPS, Jon. *Learning PHP and My SQL*. Online. 2nd edition. O'Reilly, 2007. Dostupné z: https://www.academia.edu/6811915/Learning_PHP_and_My_SQL. [cit. 2023-12-27].

DRAW PLANET. *Figma: co to je a jaké jsou jeho základní funkce a principy? - Draw Planet - výtvarné ateliéry*. Online. DRAW PLANET. DrawPlanet : kreslit a malovat dokáže každý. 2023. Dostupné z: https://www.drawplanet.cz/co-je-tedy-figma/. [cit. 2024-03-10].

FLANAGAN, David. *JavaScript: The Definitive Guide, 7th Edition*. Online. 7th edition. O'Reilly Media, 2020. ISBN 9781491952023. Dostupné z: https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016/. [cit. 2023-12-23].

GOOGLE. *Fast load times*. Online. GOOGLE. Web.dev. Dostupné z: https://web.dev/explore/fast. [cit. 2024-01-03].

HAVERBEKE, Marijn a TANTAREANU, Madalina. *Eloquent JavaScript*. Online. 3rd edition. No Starch Press, 2018. Dostupné z: https://eloquentjavascript.net. [cit. 2023-12-24].

LINDLEY, Cody. *Front-End Developer Handbook*. Online. 1119th edition. Frontend Masters, 2023. ISBN B0C4JRQVFR. Dostupné z: https://freecomputerbooks.com/Front-End-Developer-Handbook.html. [cit. 2023-12-23].

MCFEDRIES, Paul. *Web Design Playground, Second Edition*. Online. 2nd edition. Manning, 2019. ISBN 9781633438323. Dostupné z: https://www.manning.com/books/web-design-playground-second-edition. [cit. 2023-12-23].

MOZILLA CORPORATION. *Fetch API – Web APIs | MDN*. Online. MOZILLA CORPORATION. MDN Web Docs. C1998–2024, 1 April 2023. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API. [cit. 2024-01-03].

SASS. *Sass: Documentation*. Online. SASS. Sass: Syntactically Awesome Style Sheets. C2006–2023. Dostupné z: https://sass-lang.com/documentation/. [cit. 2023-12-23].

SIMPLILEARN SOLUTIONS. *What is Full Stack Development?* Online. SIMPLILEARN SOLUTIONS. Simplilearn | Online Courses – Bootcamp & Certification Platform. C2009-2023, last modified 29 August 2023. Dostupné z: https://www.simplilearn.com/what-is-full-stack-development-article. [cit. 2023-12-23].

THIRD DOOR MEDIA, INC. *What Is SEO – Search Engine Optimization?* Online. THIRD DOOR MEDIA, INC. Search Engine Land is the most helpful authority on SEO and PPC today. C2023. Dostupné z: https://searchengineland.com/guide/what-is-seo. [cit. 2023-12-23].

WEBPACK. *Webpack*. Online. 2023. Dostupné z: https://webpack.js.org. [cit. 2023-12-27].