Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vytvoření internetového obchodu s módou

Alekseev Aleksei

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Aleksei Alekseev

Informatika

Název práce

Vytvoření internetového obchodu s módou

Název anglicky

Creating e-shop for fashion store

Cíle práce

Cílem teoretické části práce je popsat možnosti programovacího jazyka JavaScript a jeho knihovny React. Dílčím cílem je analyzovat vybrané stávající e-shopy s módou. Dále je cílem definovat základní funkce, které má e-shop plnit a vytyčit zásady správného návrhu webové aplikace.

Cílem praktické části práce je navrhnout schéma fungování webové stránky e-shopu. Následně je cílem implementovat strukturu a obsah e-shopu s módou. Následně je cílem do struktury e-shopu zavést knihovnu React.

Metodika

Metodika bakalářské práce vychází z dostupné literatury z oblasti tvorby webových stránek a programovacího jazyka JavaScript. Informace získané z této literatury budou použity jak v teoretické, tak i v praktické části. V praktické části práce bude navržen a implementován internetový obchod s oblečením.

Doporučený rozsah práce

30-60 stran

Klíčová slova

internetový obchod, uživatelské rozhraní, JavaScript, ReactJS

Doporučené zdroje informací

FLANAGAN, David. JavaScript: The definitive guide: Master the world's most-used programming language. Seventh edition. O'Reilly, 2020. ISBN 978-1491952023.

WIERUCH, Robin. The Road to React. Nezávisle vydáno, 2018. ISBN 9781720043997.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Dana Vynikarová, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 31. 10. 2022

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 24. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 03. 03. 2024

Čestné prohlášení

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

V Praze dne 15.03.2024 Alekseev Aleksei

Poděkování

Rád(a) bych touto cestou poděkoval(a) mé vedoucí Ing. Daně Vynikarové, Ph. D. za trpělivost, vnímavost a objektivitu. Také všem příbuzným a přátelům za jejich podporu

Vytvoření internetového obchodu s módou

Abstrakt

Tato práce se zaměřuje na vytvoření internetového obchodu s módními produkty, kde je kladen důraz na použití technologie JavaScript a knihovny React. V teoretické části práce jsou zkoumány možnosti, které nabízí programovací jazyk JavaScript a jeho knihovna React, a jsou identifikovány hlavní charakteristiky a principy správného návrhu webových aplikací. V praktické části diplomové práce se představuje návrh struktury a obsahu internetového obchodu, jehož sortiment se zaměřuje na módní produkty. Následně je popsán proces implementace knihovny React do struktury obchodu. Výsledkem je funkční a esteticky přitažlivý internetový obchod, využívající moderní technologie a optimalizaci uživatelského zážitku.

Klíčová slova: internetový obchod, uživatelské rozhraní, JavaScript, ReactJS

Creating e-shop for fashion store

Abstract

This work focuses on the creation of an online store with fashion products, where the emphasis is on the use of JavaScript technology and the React library. The theoretical part of the thesis explores the possibilities offered by the JavaScript programming language and its React library and identifies the main characteristics and principles of proper web application design. The practical part of the thesis presents the design of the structure and content of an online store whose product range focuses on fashion products. Subsequently, the process of implementing the React library into the structure of the store is described. The result is a functional and aesthetically appealing online store, using modern technologies and optimizing the user experience.

Keywords: online store, user interface, JavaScript, ReactJS

Obsah

1 Úvod

V dnešní digitalizované a globalizované společnosti hraje elektronický obchod klíčovou roli v poskytování spotřebitelům pohodlného a široce dostupného způsobu nakupování. S rostoucím zájmem o módu a neustále se vyvíjející technologií se stává vytvoření internetového obchodu s módou zásadním krokem pro obchodníky, kteří chtějí oslovit moderní a digitálně orientovanou klientelu.

Zvýšený důraz na online nákupy módních produktů vytváří výzvu pro obchodníky, aby nejen vytvořili esteticky přitažlivý a uživatelsky přívětivý e-shop, ale také aby využili moderní technologie ke zlepšení zákaznického zážitku a efektivní správě obchodních procesů. Tato práce si klade za cíl přispět k pochopení klíčových aspektů a kroků spojených s vytvořením internetového obchodu s módou, a tím poskytnout ucelený pohled na tuto dynamickou oblast elektronického obchodu.

Tato práce se věnuje hlavním pilířům vývoje webových aplikací, s důrazem na zásady správného návrhu, vývoj v HTML/CSS, programovací jazyk JavaScript a moderní knihovnu ReactJS. Každá z těchto částí představuje kritický kámen v konstrukci efektivních a uživatelsky přívětivých webových aplikací. První část práce se zaměří na zásady správného návrhu webových aplikací, kde budeme analyzovat klíčové prvky pro vytvoření uživatelsky orientovaných, intuitivních a efektivních rozhraní. Druhá část nás provede do světa HTML/CSS, základních technologií pro strukturu a vizuální prezentaci webových stránek. Následující kapitola se bude věnovat programovacímu jazyku JavaScript, jenž je srdcem moderního webového vývoje, a konečně se zaměříme na ReactJS, knihovnu, která umožňuje vytvářet efektivní a dynamická uživatelská rozhraní. Tímto průvodcem klíčovými aspekty vývoje webových aplikací následující práce směřuje k poskytnutí hlubšího vhledu a praktických dovedností pro úspěšný a moderní webový vývoj.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem teoretické části práce je popsat možnosti programovacího jazyka JavaScript a jeho knihovny React. Dílčím cílem je analyzovat vybrané stávající e-shopy s módou. Dále je cílem definovat základní funkce, které má e-shop plnit a vytyčit zásady správného návrhu webové aplikace.

Cílem praktické části práce je navrhnout schéma fungování webové stránky e-shopu. Následně je cílem implementovat strukturu a obsah e-shopu s módou. Následně je cílem do struktury e-shopu zavést knihovnu React.

2.2 Metodika

Metodika bakalářské práce vychází z dostupné literatury z oblasti tvorby webových stránek a programovacího jazyka JavaScript. Informace získané z této literatury budou použity jak v teoretické, tak i v praktické části. V praktické části práce bude navržen a implementován internetový obchod s oblečením.

3 Teoretická východiska

3.1 Zásady správného návrhu webové aplikace

3.1.1 Uživatelské rozhraní

Existuje mnoho typů rozhraní. Rozlišují se podle povahy vzájemně komunikujících systémů, podle specifik implementace a podle svých možností. Jedno rozhraní umožňuje používat celou softwarovou část počítače. Současně jsou k plnohodnotnému používání zapotřebí speciální dovednosti. Jiné naopak umožňuje přístup pouze k omezeným funkcím, ale vyznačuje se snadným používáním.

Obvykle mluvíme o UI (user interface) neboli uživatelském rozhraní, které je nezbytné pro zajištění interakce mezi člověkem a hardwarovými a softwarovými prvky počítačového systému. Umožňuje kontakt s programy a operačními systémy (OS), které je řídí. Uživatelské rozhraní se realizuje pomocí následujících prvků:

- klávesnice;
- počítačové myši;
- joystick;
- displej;
- stylus.

Nejběžnější prvky rozhraní: tlačítko, odkaz, ikona, karta, zaškrtávací políčko, rádiové tlačítko, přepínač, rozevírací seznam, posuvník, vstupní pole, tabulky, menu. Vývoj uživatelských rozhraní je založen na atomickém designu. Jedná se o metodu diferenciace systému libovolné úrovně složitosti na jednotlivé části – atomy. V takovém případě je lze použít několikrát a vzájemně je kombinovat. [1]

- **Atomy** jsou nejmenší součásti uživatelského rozhraní: tlačítka, vstupní pole, zaškrtávací políčka, přepínače, styly pro typografii.
- **Molekuly** jsou kombinací několika atomů. Například tlačítko a vstupní pole.
- **Organismy** jsou skupiny molekul. Například navigace od konce ke konci v záhlaví nebo postranním panelu webu.

[1] MDN WEB DOCS. *User interface*. Online. Dec 18, 2023. Dostupné také z: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/user_interface. [cit. 2024-03-15].

- **Šablony** – kombinace několika organismů. Jsou to rozhraní, která umožňují provádět typické úlohy.

Fáze vývoje uživatelského rozhraní

1. **Analýza** – počáteční fází tvorby projektu je jeho koncepce. Je třeba shromáždit informace a odpovědět na nejdůležitější otázky. Musíte vědět, kdo bude mít z vašeho produktu prospěch a co může uživateli dát. V této fázi se analyzují informace o uživatelích, jejich přáních, životním stylu, zvycích, profesi. Vývojáři by měli pochopit, co potřebuje spotřebitel a co potřebuje firma.[2]
2. **Prezentace rozhraní optikou designu UX/UI** – po zkombinování získaných informací získají specialisté uživatelský scénář, z něhož vytvoří koncepci rozhraní. V této fázi vzniká základní logika vyvíjeného produktu. Později ji rozpracuje designér UX. User Experience realizuje uživatelský scénář. Podrobně popisuje všechny možné varianty chování uživatele při práci s rozhraním. Návrh bude úspěšný, pokud UX-designér dokáže uživatele skutečně pochopit a předvídat jeho jednání.[3]
3. **Vytváření prototypů** – v některých případech se prototypy vztahují k wireframům, na kterých pracují návrháři UX, a někdy se jedná o interaktivní prototypy, které fungují téměř jako hotové produkty. Tato fáze je nesmírně důležitá. Je třeba brát vážně chyby a včas je odstranit. V opačném případě budete muset vynaložit mnoho úsilí a peněz na opravu chyb v hotovém produktu.[4]
4. **Opětovná analýza** – design uživatelského rozhraní by měl být uživatelsky přívětivý a přehledný. Čím jednodušší, tím lepší. Je důležité, aby nebylo přeplněno nadměrným množstvím prvků. Spotřebitel potřebuje splnit své úkoly a nic víc. Pro kvantitativní analýzu postačí nástroj Google Analytics. Můžete provést kvalitativní výzkum a požádat člověka, aby pomocí rozhraní provedl nejjednodušší úkol. [5]

---

[2] TIDWELL, Jenifer; BREWER, Charles a VALENCIA, Aynne. *Designing Interfaces*. 3rd Edition. O'Reilly Media. ISBN 9781492051961. [cit. 2024-03-15].
[3] TIDWELL, Jenifer; BREWER, Charles a VALENCIA, Aynne. *Designing Interfaces*. 3rd Edition. O'Reilly Media. ISBN 9781492051961. [cit. 2024-03-15].
[4] TIDWELL, Jenifer; BREWER, Charles a VALENCIA, Aynne. *Designing Interfaces*. 3rd Edition. O'Reilly Media. ISBN 9781492051961. [cit. 2024-03-15].
[5] TIDWELL, Jenifer; BREWER, Charles a VALENCIA, Aynne. *Designing Interfaces*. 3rd Edition. O'Reilly Media. ISBN 9781492051961. [cit. 2024-03-15].

3.1.2 **Responsivní design a optimalizace pro různá zařízení**

Uživatelé již dlouho používají k přístupu na internet nejen domácí nebo kancelářské osobní počítače. Díky rozvoji technologií je možné používat internet téměř odkudkoli na světě a mít po ruce jedno ze zařízení – počítač, notebook, tablet nebo chytrý telefon. Všechna tato zařízení mají odlišné vlastnosti: výkon osobního počítače je vyšší než výkon chytrého telefonu a velikost širokoúhlého monitoru je několikanásobně větší než velikost tabletu. Z tohoto důvodu se zobrazení stejné webové stránky může lišit v závislosti na zařízení, na kterém je zobrazena. To lze pozorovat u starších webů, které byly spuštěny dávno před érou mobilních zařízení a od té doby nebyly modernizovány. Zobrazení takového webu na displeji chytrého telefonu je téměř nemožné, protože pravá strana stránky je odříznutá – a informace lze v nejlepším případě přečíst zvětšením a posunutím obrázku. Aby bylo prohlížení webu stejně pohodlné z jakéhokoli typu zařízení, používá se při vývoji adaptivní design nebo se vytváří mobilní verze webu.[6]

Optimalizace pro různá zařízení má řadu výhod a je důležitá pro zajištění použitelnosti webových stránek.

1. Stále více lidí používá k přístupu na internet chytré telefony a tablety. Stránky, které nejsou optimalizované, se mohou zobrazovat nesprávně a způsobovat uživatelům nepříjemnosti. Adaptivní nebo samostatný mobilní web zajistí, že interakce uživatele s webem bude pohodlná a intuitivní.[7]
2. Mobilní nebo adaptivní design umožňuje optimální vizuální a funkční interakci s webem. To zahrnuje snadnou navigaci, snadno čitelný obsah, uživatelsky přívětivé ovládání, přizpůsobené obrázky a další funkce, které zvyšují uživatelský komfort a činí web atraktivnějším pro uživatele mobilních zařízení.[8]
3. Uživatelsky přívětivý design může výrazně zvýšit konverze a prodeje. Pokud mohou uživatelé snadno a bez nepohodlí procházet a nakupovat produkty nebo služby na mobilních zařízeních, je pravděpodobnější, že nákup uskuteční. Přizpůsobivý design také usnadňuje provádění plateb a vyplňování formulářů na mobilních zařízeních, což pomáhá zvyšovat konverze.

---

[6] WEB.DEV. *Learn Responsive Design*. Online. Dostupné z: https://web.dev/learn/design. [cit. 2024-03-15].
[7] WEB.DEV. *Learn Responsive Design*. Online. Dostupné z: https://web.dev/learn/design. [cit. 2024-03-15].
[8] WEB.DEV. *Learn Responsive Design*. Online. Dostupné z: https://web.dev/learn/design. [cit. 2024-03-15].

4. Vyhledávače zohledňují přívětivost webových stránek pro mobilní zařízení při řazení výsledků vyhledávání. Mobilně přívětivý design pomáhá zlepšit pozici webu ve výsledcích vyhledávání na mobilních zařízeních, což může vést ke zvýšení návštěvnosti a počtu návštěvníků. [9, 10]

3.1.3 O principu Mobile-first

Strategie Mobile First, jak už název napovídá, zahrnuje navrhování rozhraní webových stránek primárně pro mobilní zařízení. Filozofie tohoto přístupu spočívá v tom, že se nejprve vytvoří prototyp návrhu pro nejmenší obrazovku a poté vývojáři přejdou na větší displeje. To je hlavní rozdíl oproti klasickému přístupu Desktop First, který weboví specialisté používají již řadu let.

Tradičně proces tvorby webových stránek začíná vývojem vzhledu pro počítače a notebooky, protože mají vysoký výkon a na velkou obrazovku lze umístit velké množství obsahu. Poté se hotová verze přizpůsobí pro ostatní zařízení s ohledem na jejich možnosti. Proč jste tedy museli přijít s konceptem, kde vše probíhá v opačném pořadí?[11]

Jde o to, že ne všechny vizuální a funkční prvky stránek pro stolní počítače lze přenést do mobilní verze při zachování stejně snadné interakce. Poměrně často optimalizaci webu pro chytré telefony provázejí různé problémy: dlouhé načítání stránek, složitá navigace, nepohodlná tlačítka CTA(Call-to-action) atd. Pro malou obrazovku chytrého telefonu musíte odstranit značnou část krásných, ale nefunkčních prvků. Stránky, které jsou okamžitě optimalizovány pro mobilní zařízení, se načítají rychleji než stránky adaptivní. Lehkost webu navrženého podle zásad Mobile First zajišťují[12]:

- Umístění pouze nejdůležitějšího obsahu
- Použití obrázků bez ztráty kvality, ale s menší velikostí
- Absencí těžkopádného kódu, kterého se dosáhne změnou velikosti prvků webu pomocí CSS a množstvím mediálních dotazů

---

[9] MDN. *Responsive design*. Online. Jan 8, 2024. Dostupné také z: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design. [cit. 2024-03-15].
[10] LEVY, Jaime. *UX-Strategy*. O'Reilly Media. ISBN 978-1-449-37286-6. [cit. 2024-03-15].
[11] MICHÁLEK, Martin. *Co je to „Mobile First“? Ale doopravdy*. Online. 22.8.2015. Dostupné z: https://www.vzhurudolu.cz/prirucka/mobile-first. [cit. 2024-03-15].
[12] MICHÁLEK, Martin. *Co je to „Mobile First“? Ale doopravdy*. Online. 22.8.2015. Dostupné z: https://www.vzhurudolu.cz/prirucka/mobile-first. [cit. 2024-03-15].

- Eliminace případných "berliček", které mohou programátoři použít pro přizpůsobení

Jedna nebo dvě sekundy – tak dlouho je většina lidí ochotna čekat na takzvaný First contentful paint (FCP). I když uživatele zajímá popis stránky ve vyhledávači, hrozí, že opustí web, který se dlouho načítá. [13, 14]

3.1.4 Zabezpečení webu a ochrana osobních údajů

Existuje několik základních způsobů ochrany webu:

- Zajistit ochranu před útoky DDoS.
- Aktivace certifikátu SSL.
- Používat spolehlivý hosting.
- Používat bezpečné moduly plug-in/knihovny/frameworky/CMS.
- Používat stávající techniky ochrany proti útokům typu SQL injection a XSS.
- Zajistit protokolování a monitorování bezpečnostních událostí na webových stránkách.
- Pravidelně zálohovat webové stránky a všechna důležitá data.
- Používat silná a složitá hesla a ochranu heslem.
- Pokud je k dispozici panel pro správu obsahu webových stránek, změňte výchozí. přihlašovací adresu a zajistěte kontrolu přístupu.

Každá položka má samozřejmě své vlastní "ale" a řadu dílčích položek, které by měly být zdůrazněny. Lze je také rozdělit do podskupin na základě následujících úvah: některé činnosti vyžadují jednorázové připojení, konfiguraci a občasnou kontrolu výkonu (hosting a konfigurace certifikátu SSL), zatímco jiné zahrnují neustálé kontroly, aktualizace a vyžadují zvýšenou pozornost (vše ostatní).[15]

---

[13] Google. Online. *Find out how you stack up to new industry benchmarks for mobile page speed*. Feb 2018. Dostupné z: https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks/. [cit. 2024-03-15].
[14] MDN. *Mobile First*. Online. Jan 8, 2023. Dostupné také z: https://developer.mozilla.org/en-US/docs/Glossary/Mobile_First. [cit. 2024-03-15].
[15] HOFFMAN, Andrew. *Web Application Security: Exploitation and Countermeasures for Modern Web Applications*. O'Reilly Media. ISBN 978-1492053118. [cit. 2024-03-15].

3.1.5 Testování a ladění aplikace

Testování webových stránek je poslední a povinnou fází technického vývoje webových stránek. V procesu tvorby zdroje hraje klíčovou roli, protože právě kvalita testování rozhoduje o budoucí životnosti zdroje. Vždyť zdroj, který obsahuje chyby, způsobuje negativní reakce návštěvníků a v důsledku toho jejich ztrátu. V důsledku toho musí vlastník zdroje zaplatit za revizi a někdy i za opětovný vývoj zdroje. Testování webu může zabrat až 50 % času a rozpočtu. Abychom pochopili, kde se takové číslo bere, v tomto článku si povíme, jak probíhá testování webových stránek, abychom na konci získali kvalitní produkt.[16]

Účelem testování je obecná kontrola skutečného fungování webových stránek z hlediska souladu s požadavky. Celá fáze je pečlivou prací specialistů, kteří za účelem odhalení chyb vytvářejí umělé situace, které mohou nastat při provozu prostředku, a analyzují "chování" prostředku za navržených podmínek. Po identifikaci chyb (omylů) tester vyhotoví zprávu a předá ji vedoucímu projektu, který práci na jejich odstranění rozdělí mezi účastníky projektu. Po revizi je prostředek znovu otestován. Takový cyklus prací se opakuje, dokud webová stránka nedosáhne při testování stanovených výsledků.

Testování webových stránek na chyby se provádí různými metodami v různých prohlížečích. Aby bylo dosaženo bezchybného fungování webových stránek, vytvoří odborníci na testování akční plán, který zahrnuje:

- **Testování funkčnosti** – jedná se o klíčovou fázi testování, kterou nelze nahradit ani přeskočit. Chyba ve funkčnosti nevyhnutelně povede ke ztrátě zákazníků. Pokud například dojde k chybě ve funkci nákupního košíku v internetovém obchodě, návštěvník nebude moci produkt zakoupit a kvůli chybě ve vyplnění formuláře vstupní stránky se jednostránkový web nikdy nepohne z nulového konverzního poměru. Nejčastěji se při funkčním testování kontrolují: Navigace, vyhledávání a objednávání, nákup a platba zboží, Registrační formulář nebo přihlášení do osobní skříně, Možnosti úprav webu (přidávání pozic, úprava obsahu atd.).[17]

---

[16] HAMILTON, Thomas. *Web Application Testing: How to Test a Website?* Online. 09.03.2024. Dostupné z: https://www.guru99.com/web-application-testing.html. [cit. 2024-03-15].
[17] HAMILTON, Thomas. *Web Application Testing: How to Test a Website?* Online. 09.03.2024. Dostupné z: https://www.guru99.com/web-application-testing.html. [cit. 2024-03-15].

- **Testování použitelnosti(usability)** - testování použitelnosti zdrojů by mělo ukázat, jak je uživateli jasné, jak produkt nebo službu najít a objednat, zda je pro návštěvníka pohodlné být na stránce, pohybovat se mezi nimi atd. Tester objektivně hodnotí projekt a identifikuje chyby, které mohou způsobit nepohodlí, a v důsledku toho přimět člověka zavřít kartu se stránkou. Testování použitelnosti stránek přímo ovlivňuje vnímání firmy návštěvníkem. Pokud je mu pobyt na stránce nepříjemný, nikdy se na ni nevrátí.[18]
- Testování výkonnosti – v této fázi se analyzuje a testuje výkonnost zdroje. Předpokládá se, že odborník pochopí, zda stránky vydrží zátěž velkého počtu uživatelů, pokud ve stejný okamžik provádějí nějaké akce. Testování se provádí automaticky, uměle se vytvoří maximální zatížení zdroje a sledují se výsledky.[19]
- Testování bezpečnosti – kvalita zabezpečení zdroje určuje jeho spolehlivost. Hlavním účelem této metody testování je identifikovat zranitelnosti webu při různých útocích.[20]
- Testování uživatelského rozhraní – tento typ testování zahrnuje kontrolu splnění požadavků na grafické rozhraní: zda obsahuje jednotný styl, jak profesionálně vypadá. Ve stejné fázi se kontroluje také cross-browser (správné zobrazení webu ve všech existujících prohlížečích bez ohledu na verzi), adaptivní verze (pokud je poskytována).

3.2 HTML / CSS

HTML a CSS jsou dva hlavní nástroje, které potřebujete při práci se šablonami webových stránek:

---

[18] HAMILTON, Thomas. *Web Application Testing: How to Test a Website?* Online. 09.03.2024. Dostupné z: https://www.guru99.com/web-application-testing.html. [cit. 2024-03-15].
[19] HAMILTON, Thomas. *Web Application Testing: How to Test a Website?* Online. 09.03.2024. Dostupné z: https://www.guru99.com/web-application-testing.html. [cit. 2024-03-15].
[20] HAMILTON, Thomas. *Web Application Testing: How to Test a Website?* Online. 09.03.2024. Dostupné z: https://www.guru99.com/web-application-testing.html. [cit. 2024-03-15].

- **HTML** je standardizovaný programovací jazyk, který funguje jako značkovací jazyk. Využívá značky k identifikaci různých typů obsahu a určení jejich účelu na webových stránkách.
- **CSS** je zkratka pro kaskádové styly. Jedná se o značkovací jazyk, který definuje, jak mají být prvky HTML na webových stránkách zobrazeny na rozhraní stránky.

Zatímco tedy jazyk HTML slouží k popisu toho, jaké informace se mají na stránce zobrazit a v jakém pořadí, CSS rozšiřuje možnosti jazyka HTML a umožňuje měnit barvy, písma, pozadí atd.

3.2.1 Základy HTML pro strukturu webových stránek

Webová stránka je textový soubor s příponou .html. Obsahuje veškerý obsah stránky, včetně textu, značek atd. Uvnitř každého takového souboru musí být několik povinných značek, jako např:

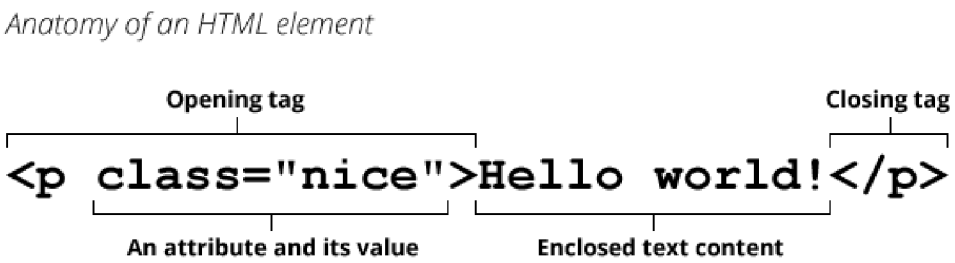
- <!DOCTYPE html> na začátku dokumentu.
- <html> na začátku a </ html> na konci dokumentu – mezi těmito dvěma značkami by měl být celý obsah stránky se všemi značkami (vše mimo bude ignorováno).
- <head> na začátku a </head> na konci výčtu servisních značek.
- <title> je servisní značka, je s ní zvýrazněn název stránky (karta v prohlížeči bude podepsána)
- <meta> je servisní značka, která určuje kódování stránky (obvykle utf-8).
- <body> na začátku a </ body> na konci obsahu stránky – obsah stránky, texty, značky odstavců, nadpisy atd. jsou již zobrazeny mezi těmito dvěma značkami.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Nazev webu</title>
  </head>
  <body>
    Zde by se měl nacházet obsah stránky.
  </body>
</html>
```

Obrázek 1 Struktura webu

Hlavními složkami HTML jsou značky a atributy:

- **Značka (tag)** je přesně to, co je uzavřeno v úhlových závorkách, jako je <title>. Nejčastěji existují značky v páru – otevírání a zavírání. Otvírací značka je umístěna na začátku prvku a uzavírací, respektive na konci. Tímto způsobem vymezujeme hranice akce značky.
- **Atribut** – doplňkové vlastnosti připisované uvnitř značky přiřadit prvku (textu, obrázku atd.) jakékoli informace. Všechny atributy se skládají ze dvou částí-je to název a hodnota



Obrázek 2 Anatomie elementu HTML

3.2.2 Stylování webových stránek pomocí CSS

Zkratka CSS znamená Cascading Style Sheets, což je značkovací jazyk používaný pro vizuální design webových stránek.

Zatímco jazyk HTML je zodpovědný za rozvržení objektů na stránce, CSS určuje jejich vzhled, včetně velikosti, barvy, obrázku na pozadí, stupně průhlednosti, polohy vůči ostatním prvkům, chování při najetí na stránku a vizuální změny tlačítek po kliknutí. Jazyk CSS se rychle stal standardem ve vývoji webu, protože umožňuje rychlou změnu vizuálního designu webu, aniž by se uchýlil k použití složitějších programovacích jazyků. Syntaxe CSS v samostatném souboru v příslušném formátu (.CSS) vypadá takto: *volič {vlastnost: hodnota;}*

- **Volič** (selektor) je odkaz na prvek v HTML, na kterém bude provedena práce (dekorace)
- **Vlastnost** je specifická vlastnost prvku, kterou je třeba změnit. Například velikost nebo barva.
- **Hodnota** je digitální nebo textové označení pro vybranou vlastnost.

Kromě toho je jedním z nejdůležitějších nástrojů při vytváření adaptivního pracovního webu použití dotazů na média (media query). Mediální dotazy jsou speciální podmíněné konstrukce, které umožňují použití stylů pouze pro určitá zařízení.

Mediální dotazy jsou zaznamenány následovně: *@Media (podmínky) {/*CSS kód, který bude pro tuto podmínku splněn */}*

Jako podmínky mohou fungovat různé hodnoty a konstanty.

3.3 Programovací jazyk JavaScript

JavaScript je univerzální programovací jazyk, významně ovlivňující oblast webového vývoje. Tento skriptovací jazyk umožňuje tvorbu interaktivních a dynamických webových stránek, a to přímo v prohlížeči uživatele. Jeho objektově orientovaná povaha usnadňuje strukturování kódu a manipulaci s obsahem stránky. JavaScript reaguje na uživatelské akce, což přispívá k vytváření plynulých uživatelských rozhraní. Jazyk poskytuje možnost asynchronní komunikace se serverem, což vede ke zvýšení efektivity a rychlosti webových aplikací. Díky podpoře mnoha knihoven a frameworků, jako jsou React, Angular a Vue, JavaScript usnadňuje vývoj komplexních webových projektů. Moderní verze jazyka přinášejí nové funkce a vylepšení, což zvyšuje jeho konkurenceschopnost v rámci programování. S aktivní komunitou vývojářů a množstvím dostupných zdrojů a tutoriálů pro učení se, JavaScript se stal klíčovým nástrojem pro vývoj moderních webových aplikací. Navíc není omezen pouze na frontend vývoj, ale je také používán pro serverový vývoj pomocí Node.js, což umožňuje sjednocení jazyka na celém webovém stacku. Jeho otevřený přístup kódování a schopnost rychle reagovat na nové technologické trendy zdůrazňují jeho význam v současném digitálním prostředí.

3.3.1 Historie a vývoj JavaScript

První předpoklady pro vznik tohoto jazyka se objevily již v roce 1992, kdy byl zahájen vývoj skript vloženého jazyka Cmm (C mínus mínus). Později byl přejmenován na ScriptEase, protože název „C mínus mínus“ měl negativní nádech. Než jazyk získal své moderní jméno, jeho název se ještě několikrát změnil.

V roce 1995 dostal Brendan Eich za úkol implementovat programovací jazyk do prohlížeče Netscape. Zpočátku se jazyk jmenoval Mocha, poté LiveScript. Nakonec dostal své moderní jméno – JavaScript. Zde se vývojáři pustili do triku. V době, kdy se zabývali

vylepšováním Livescriptu, byl jazyk Java poměrně oblíbený. Aby bylo možné získat více vývojářů pro práci s novým jazykem, bylo rozhodnuto použít v jeho názvu Javu. Nakonec to byl JavaScript:

„***InfoWorld:*** *As I understand it, JavaScript started out as Mocha, then became LiveScript and then became JavaScript when Netscape and Sun got together. But it actually has nothing to do with Java or not much to do with it, correct?*

Eich: *That's right. It was all within six months from May till December (1995) that it was Mocha and then LiveScript. And then in early December, Netscape and Sun did a license agreement and it became JavaScript. And the idea was to make it a complementary scripting language to go with Java, with the compiled language.* " [21]

Zatím poslední verze jazyka ES6 vyšla v roce 2015. S jejím vznikem získal jazyk druhý život. Objevily se nové normy a také možnost práce z konstanty. Změnil se i samotný kód. Jazyk dodržuje princip redukce kódu při větší funkčnosti. Mezi hlavní vlastnosti tohoto programovacího jazyka patří:

- Dynamické typování. To znamená, že datový typ se určí pouze tehdy, když je proměnné nebo konstantě přiřazena její hodnota.
- Flexibilní práce s funkcemi. V JS můžete funkce nejen vykonávat, ale také z nich vracet, předávat funkce jako parametry jiným funkcím a přiřazovat funkce jako hodnoty proměnným.
- Jazyk JavaScript je podporován všemi moderními prohlížeči.
- Objektově orientované programování. Jedná se o metodiku programování, ve které je celý program reprezentován jako sada objektů.

Kromě toho je důležitou vlastností JavaScript jeho pokročilá infrastruktura. Vývojáři dnes mohou využívat velké množství knihoven a frameworků (mezi nejpopulárnější patří React, Angular a Vue), několik nástrojů pro tvorbu stránek, pomocných knihoven (např. Lodash) a generátorů statických stránek.

Co se týče oblastí použití, v oblasti vývoje webových stránek se především hojně využívá jazyk JavaScript. A funguje v kombinaci s jazyky HTML a CSS. Pomocí JS lze vytvářet libovolné aplikace v prohlížeči.

---

[21] KRILL, Paull (ed.). JavaScript creator ponders past, future. Online. *InfoWorld*. S. 1. Dostupné z: https://www.infoworld.com/article/2653798/javascript-creator-ponders-past-future.html. [cit. 2024-03-15].

3.3.2 Základní syntaxe a principy programování

Není to tak dávno, co se objevila nová generace aplikací v JavaScript, které se nijak neliší od desktopových aplikací – neuvěřitelný pokrok v oblasti webových technologií. Pryč jsou pomalé požadavky na stránku při každé interakci uživatele s aplikací. Motory JavaScript se staly tak výkonnými, že je nyní možné ukládat stav na straně klienta, což výrazně urychlilo odezvu aplikace a zlepšilo její kvalitu.[22]

Pokud znáte jiné programovací jazyky, možná se vám bude hodit informace, že JavaScript je vysokoúrovňový, dynamický, netypovaný a interpretovaný programovací jazyk, který se dobře hodí pro objektově orientovaný a funkcionální styl programování. JavaScript zdědil syntaxi od Javy, své funkce od Scheme a mechanismus dědičnosti založený na prototypech od Selfu.[23]

3.3.2.1 Symboly

Při psaní programů v jazyce JavaScript se používá znaková sada Unicode. Unicode je nadmnožinou kódování ASCII a Latin-1 a podporuje prakticky všechny psané jazyky na světě. Standard ECMAScript 3 vyžaduje, aby implementace jazyka JavaScript podporovaly kód Unicode verze 2.1 nebo vyšší, a standard ECMAScript 5 vyžaduje, aby implementace podporovaly kód Unicode verze 3 nebo vyšší. [24]

3.3.2.2 Citlivost velkých a malých písmen

Jazyk JavaScript rozlišuje malá a velká písmena. To znamená, že klíčová slova, názvy proměnných a funkcí a další identifikátory jazyka musí vždy obsahovat stejnou sadu velkých a malých písmen.

Například klíčové slovo while by se mělo psát jako "while", nikoli "While " nebo "WHILE". Podobně myvar, Myvar, MyVar a MYVAR jsou názvy čtyř různých proměnných. Všimněte si však, že značkovací jazyk HTML (na rozdíl od XHTML) nerozlišuje velká a malá písmena. Vzhledem k tomu, že jazyky HTML a JavaScript na straně klienta spolu úzce souvisejí, může být toto rozlišování matoucí. Mnoho objektů jazyka

---

[22] W3 SCHOOLS. *JavaScript Syntax*. Online. Dostupné z: https://www.w3schools.com/js/js_syntax.asp. [cit. 2024-03-15].
[23] FLANAGAN, David. Lexical Structure. In: *JavaScript: The definitive guide: Master the world's most-used programming language.*. Seventh Edition. O'Reilly. ISBN 978-1491952023. [cit. 2024-03-15].
[24] MDN. *Lexical grammar* [online]. Nov 8, 2023. Dostupné také z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar [cit. 2024-03-15].

JavaScript a jejich vlastností má stejné názvy jako značky a atributy jazyka HTML, které označují. Zatímco však v jazyce HTML lze tyto značky a atributy psát s libovolným počátečním písmenem, v jazyce JavaScript se obvykle musí psát malými písmeny.[25]

Například atribut onclick obsluhy události se v jazyce HTML nejčastěji uvádí jako onClick, ale v kódu jazyka JavaScript (nebo v dokumentu XHTML) musí být označen jako onclick. [26]

3.3.2.3 **Mezery, posuvy řádků a řídicí znaky formátu**

JavaScript ignoruje mezery, které mohou být v programu mezi tokeny. Kromě toho JavaScript z velké části ignoruje také znaky pro posuv řádků. Proto lze ve zdrojovém kódu programu bez omezení používat mezery a znaky pro posuv řádků, aby byl kód formátován a čitelný.[27]

3.3.2.4 **Nepovinné středníky**

Stejně jako v jiných programovacích jazycích se v jazyce JavaScript používají středníky (;) k oddělení jednotlivých instrukcí. Používání středníků je důležité, aby byly zřejmé úmysly programátora: bez tohoto oddělovače by bylo možné omylem zaměnit konec jedné instrukce za začátek další a naopak.

Typicky v jazyce JavaScript není středník mezi instrukcemi potřeba, pokud jsou na různých řádcích. (Středník můžete vynechat také na konci programu nebo v případě, že dalším tokenem v programu je uzavírací závorka}). Mnoho programátorů v JavaScript používá středníky k výslovnému označení konců instrukcí, i když to není nutné.[28]

Podívejte se na následující úryvek. Protože se obě instrukce nacházejí na různých řádcích, lze první středník vynechat:

```
1    a = 3
2    b = 5;
```

Obrázek 3

---

[25] W3 SCHOOLS. *JavaScript Syntax*. Online. Dostupné z: https://www.w3schools.com/js/js_syntax.asp. [cit. 2024-03-15].
[26] MDN. *Grammar and types* [online]. Feb 11, 2024. Dostupné také z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Grammar_and_types [cit. 2024-03-15].
[27] W3 SCHOOLS. *JavaScript Syntax*. Online. Dostupné z: https://www.w3schools.com/js/js_syntax.asp. [cit. 2024-03-15].
[28] W3 SCHOOLS. *JavaScript Syntax*. Online. Dostupné z: https://www.w3schools.com/js/js_syntax.asp. [cit. 2024-03-15].

Pokud jsou však tyto pokyny zapsány tak, jak je uvedeno níže, je první středník povinný:

```
1    a = 3; b = 5;
```

Obrázek 4

3.3.2.5 **Identifikátory a vyhrazená slova**

Identifikátor je jednoduše název. V jazyce JavaScript slouží identifikátory jako názvy proměnných a funkcí a také jako označení některých cyklů. Identifikátory musí začínat písmenem, podtržítkem (_) nebo znakem dolaru ($). Za nimi mohou následovat libovolná písmena, číslice, podtržítka nebo znaky dolaru. (Číslo nemůže být prvním znakem, protože pak by interpret obtížně rozlišoval identifikátory od čísel.) Příklady platných identifikátorů: „i“, „variable_name“, „_123“, „$abc123“ [29]

Z důvodu kompatibility a snadné editace se pro sestavení identifikátorů obvykle používají pouze znaky ASCII a čísla. JavaScript však umožňuje používat v identifikátorech písmena a číslice z celé znakové sady Unicode. To umožňuje programátorům pojmenovávat proměnné v jejich rodném jazyce a používat v nich matematické symboly

Kromě toho je v jazyce JavaScript vyhrazena řada identifikátorů, které fungují jako klíčová slova samotného jazyka. Tato klíčová slova nemohou sloužit jako identifikátory v programech. JavaScript si také vyhrazuje některá klíčová slova, která v současné době nejsou součástí jazyka, ale která se mohou stát jeho součástí v budoucích verzích.

Je důležité vědět, že konkrétní implementace jazyka JavaScript mohou obsahovat vlastní předdefinované globální proměnné a funkce. Kromě toho může mít každá konkrétní platforma JavaScript (klient, server a další) svůj vlastní seznam globálních vlastností.[30]

3.3.3 **Frameworky a knihovny JS**

V dnešní době existuje v oboru IT příliš mnoho programovacích jazyků. Některé se používají pro vývoj front-endu, jiné pro vývoj back-endu. Existuje jen několik jazyků, které lze použít pro obojí. Navzdory tomu se JavaScript stále rozšiřuje.

---

[29] MDN WEB DOCS. *Identifier*. Online. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/Identifier. [cit. 2024-03-15].
[30] FLANAGAN, David. Identifiers and Reserved Words. In: *JavaScript: The definitive guide: Master the world's most-used programming language.*. Seventh Edition. O'Reilly. ISBN 978-1491952023. [cit. 2024-03-15].

Když se však mluví o knihovnách JavaScript vs. frameworcích JavaScript, bývají tyto termíny zaměňovány, přičemž knihovnám se říká frameworky a frameworkům knihovny.

K dosažení požadované funkčnosti a výkonu můžete potřebovat kombinaci různých nástrojů. Ne každý projekt však vyžaduje použití konkrétního frameworku nebo knihovny. Pochopení hloubky ekosystému JavaScript a nalezení inovativních nástrojů vám umožní vytvářet aplikace efektivněji.

3.3.3.1 **JavaScript Framework**

Framework JavaScript je sada knihoven kódu JS, které poskytují vývojářům webových aplikací předem napsaný kód pro každodenní programování. Rámce mají specifický kontext a pomáhají při vytváření webových aplikací v tomto kontextu. Poskytují vám velkou kontrolu nad vaší aplikací. Mohou vás také nasměrovat správným směrem, pokud jde o architekturu a následný projekt.[31]

Frameworky se skládají z mnoha knihoven, které obsahují háčky a zpětná volání, takže na nich můžete stavět. Vývoj navíc pohání zvýšený požadavek na jednoduchost a design a také neustálá aktualizace JS frameworků.

Populární frameworky:

- **Angular**

  Angular je kompletní framework JavaScript pro tvorbu webových aplikací vyvinutý společností Google v roce 2010 (Angular JS). Od té doby se Angular výrazně změnil a byl vydán v několika verzích. Nabízí širokou škálu funkcí a nástrojů pro vytváření škálovatelných, udržovatelných a vysoce výkonných webových aplikací. Nejnovější verzí je Angular 16.[32]

- **NestJS**

  NestJS je populární framework JavaScript určený k vytváření škálovatelných aplikací na straně serveru. V roce 2016 jej vyvinul Kamil Myslivets. Důležitými vlastnostmi NestJS jsou škálovatelnost a udržovatelnost, podporuje také

---

[31] MDN WEB DOCS. *Understanding client-side JavaScript frameworks*. Online. 05.03.2024. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks. [cit. 2024-03-15].

[32] ANGULAR. *Understanding Angular*. Online. Dostupné z: https://angular.io/guide/understanding-angular-overview. [cit. 2024-03-15].

TypeScript a využívá všechny jeho funkce pro efektivní vývoj. Nejnovější verzí je NestJS 9.[33]

- **Vue**

  Vue je progresivní framework pro vytváření uživatelských rozhraní. V roce 2014 ho vyvinul Evan Yu. Vue se zaměřuje na prezentační vrstvu aplikace a nabízí vývojářům nástroje a vzory pro vytváření interaktivních webových rozhraní. Jeho rostoucí popularitě nahrává jednoduchost a přizpůsobivost při integraci s dalšími technologiemi. Nejnovější verzí je Vue 3.[34]

3.3.3.2 Knihovny JavaScript

Knihovna JS je část opakovaně použitelného kódu s jediným primárním případem použití. Knihovna JavaScript může mít v závislosti na jazyku více funkcí/objektů/metod. Abyste měli přístup k těmto možnostem, vaše aplikace se s knihovnou propojí. Knihovny JavaScript odstraňují nutnost znovu vynalézat kolo pro určité části kódu. Výsledkem je efektivnější a bezchybný vývoj.[35]

K dispozici je přibližně 83 knihoven jazyka JavaScript, které pokrývají strojové učení, rychlé prototypování, animovanou 3 D grafiku, manipulaci s DOM a další.

Populární knihovny:

- **jQuery**

  jQuery je knihovna pro programovací jazyk JavaScript, sada nástrojů pro vývoj webových stránek. S její pomocí vývojář přistupuje k různým prvkům webové stránky a ovládá její obsah. Tímto způsobem lze měnit stránku v závislosti na nastavení a akcích uživatele. Knihovnu používají vývojáři frontendů, kteří vytvářejí viditelnou část webových stránek a webových aplikací. Často se jedná o programátory, kteří udržují starší kód v dlouhodobých projektech. [36]

- **Axios**

  Axios je populární knihovna jazyka JavaScript, která podporuje klientské i serverové prostředí. Používá se k odesílání požadavků HTTP. [37]

---

[33] NEST JS. *Documentation*. Online. Dostupné z: https://docs.nestjs.com/. [cit. 2024-03-15].
[34] VUE.JS. *Introduction*. Online. Dostupné z: https://vuejs.org/guide/introduction.html. [cit. 2024-03-15].
[35] *David Sawyer McFarland (2014). JavaScript & JQuery: The Missing Manual. O'Reilly Media. p. 106. ISBN 9781491948620.* [cit. 2024-03-15].
[36] JQUERY. *API Documentation*. Online. Dostupné z: https://api.jquery.com/. [cit. 2024-03-15].
[37] AXIOS. *What is Axios?* Online. Dostupné z: https://axios-http.com/docs/intro. [cit. 2024-03-15].

- **Lodash**

  Lodash je výkonná a široce používaná knihovna, která zjednodušuje úlohy, jako je manipulace s poli, objekty a řetězci. [38]

- **React**

3.4 Knihovna React

3.4.1 Úvod do React

React.js (nebo jen React) je JavaScript knihovna vyvinutá společností Facebook pro tvorbu uživatelských rozhraní (UI) na webových stránkách nebo webových aplikacích. React umožňuje vytvářet interaktivní UI komponenty, které reagují na uživatelské akce nebo změny v datech, což usnadňuje vytváření rychlých, efektivních a dynamických webových stránek. React se stává stále populárnější díky své jednoduchosti, efektivitě a velké komunitě, která poskytuje podporu a rozšíření.

3.4.2 Základní pojmy a zásady

3.4.2.1 Komponenty a Stav

Hlavním rysem React.js jsou komponenty a stavy.

Komponenta je kus kódu, který je zodpovědný za vzhled jednoho z prvků webu nebo aplikace. A takové kousky-komponenty mohou být vnořené.

Stav jsou všechny informace o prvku, včetně jeho zobrazení. Například stav objektu "teploměr" lze popsat pomocí vlastností current_temperature, min a max. [39]

Vzhledem ke specifikům této knihovny může kód React vypadat pro někoho, kdo píše v jazyce JavaScript, nezvykle: v tagu <body> totiž není téměř žádný layout. Zaměřme se však přímo na aplikaci počítadla: její hlavní logika se nachází na řádcích 25-40.

Veškerý kód se nachází uvnitř funkce App. V React se jí a dalším podobným funkcím říká komponenty. Komponenta je fragment rozhraní, který obsahuje značku a případně související logiku. Všechny aplikace React jsou postaveny na komponentách. Samotný

---

[38] LODASH. *Why Lodash?* Online. Dostupné z: https://lodash.com/. [cit. 2024-03-15].
[39] REACT.JS. *State: A Component's Memory* [online]. Dostupné z: https://react.dev/learn/state-a-components-memory [cit. 2024-03-15].

komponentový přístup se však objevil dávno před React, zde však byl spojen s deklarativností. [40]

index.html > ...

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Parcel Sandbox</title>
    <meta charset="UTF-8" />
  </head>

  <body>
    <div id="root"></div>

    <script
      src="https://unpkg.com/react@17/umd/react.development.js"
      crossorigin
    ></script>
    <script
      src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"
      crossorigin
    ></script>
    <script
      src="https://unpkg.com/@babel/standalone@7.13.6/babel.min.js"
      crossorigin
    ></script>

    <script type="text/babel">
      function App() {
        const [value, setValue] = React.useState(0);

        function increment() {
          setValue(value + 1);
        }

        return (
          <div className="app">
            <h1>{value}</h1>
            <button onClick={increment}>+1</button>
          </div>
        );
      }

      ReactDOM.render(
        <div>
          <App />
        </div>,
        document.getElementById("root")
      );
    </script>
  </body>
</html>
```

Obrázek 5 Struktura ReaktJS

3.4.2.2 **Virtuální DOM**

Než se dostaneme k tomu, co je virtuální DOM, musíme vysvětlit, co je skutečný DOM. DOM (zkratka pro Document Object Model) je způsob reprezentace strukturovaného dokumentu pomocí objektů. Jedná se o multiplatformní a na jazycích nezávislou konvenci pro reprezentaci a interakci s daty v jazycích HTML, XML atd.[41]

---

[40] REACT.JS. *Your First Component* [online]. Dostupné z: https://react.dev/learn/your-first-component [cit. 2024-03-15].
[41] REACT.JS. *Virtual DOM and Internals*. Online. Dostupné z: https://legacy.reactjs.org/docs/faq-internals.html. [cit. 2024-03-15].

Webové prohlížeče zpracovávají komponenty DOM a my s nimi můžeme komunikovat pomocí JavaScript a CSS. Můžeme manipulovat s uzly dokumentu, upravovat jejich data, mazat a vkládat nové uzly. V dnešní době je rozhraní DOM API do značné míry multiplatformní a multiprohlížečové. [42]

3.4.2.2.1 V čem je tedy problém?

Hlavním problémem DOM je, že nikdy nebyl navržen pro vytváření dynamického uživatelského rozhraní. Můžeme s ním pracovat pomocí jazyka JavaScript a knihoven, jako je jQuery, ale jejich použití neřeší problémy s výkonem.

Podívejte se na moderní sociální sítě, jako je Twitter, Facebook nebo Pinterest. Po troše rolování budeme mít desítky tisíc uzlů DOM, s nimiž efektivní interakce není snadný úkol.

Jako příklad si můžete zkusit posunout 1000 bloků divů o 5 pixelů doleva. To může trvat více než sekundu – to je na moderní internet příliš mnoho. Můžete optimalizovat skript a použít některé techniky, ale ve výsledku to při práci s obrovskými stránkami a dynamickým uživatelským rozhraním způsobí jen další problémy.

Jedním ze způsobů, jak tento problém vyřešit, je použít přístup Virtual DOM. Virtuální DOM není standardem a stále s DOM interagujeme, ale co nejméně často a co nejefektivněji.

3.4.2.2.2 Virtual DOM

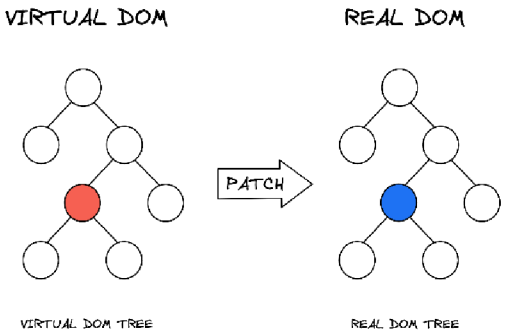
Místo přímé interakce s DOM pracujeme s jeho odlehčenou kopií. V této kopii můžeme provádět změny podle našich potřeb pomocí JSX a poté je aplikovat na skutečný DOM. Tím se porovná strom DOM s jeho virtuální kopií, zjistí se rozdíl a spustí se překreslení toho, co bylo změněno.[43]

---

[42] MDN WEB DOCS. *Document Object Model (DOM)* [online]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model [cit. 2024-03-15].
[43] REACT.JS. *Virtual DOM and Internals*. Online. Dostupné z: https://legacy.reactjs.org/docs/faq-internals.html. [cit. 2024-03-15].

Obecně lze říct, že React funguje takto:

1. Připojení způsobí vykreslení aplikace.
2. Výsledný DOM se vloží do skutečného DOM celý, protože v něm ještě nic není. Virtuální DOM je zase uložen uvnitř systému React pro pozdější aktualizaci.
3. Změna stavu způsobí výpočet nového virtuálního DOM.
4. Vypočítá se rozdíl mezi starým a novým virtuálním DOM.
5. Rozdíl se aplikuje na skutečný DOM. [44]



Obrázek 6 Jak funguje DOM

3.4.2.3 **Jednosměrný tok dat**

V React proudí data směrem od rodičovské komponenty k podřízené komponentě. To znamená, že změna dat v jedné komponentě nemá vliv na ostatní komponenty. Data jsou předávána prostřednictvím vlastností (props) a lze je měnit pouze v rodičovské komponentě.[45]

3.4.2.4 **JSX**

JSX je preprocesor, který do jazyka JavaScript přidává syntaxi XML, a proto se jmenuje JavaScript XML – JSX. Tato technologie je úvodem do samotné technologie React a poskytuje představu o struktuře aplikací React. Ti, kdo se s JSX setkávají poprvé, si myslí, že jde o podivnou směs HTML a JavaScript, ale to není tak docela pravda, spíše vůbec. Vývojáři JSX vytvořili syntaxi, která nám umožňuje popisovat uživatelská rozhraní ve známém frontendovém stylu podobném HTML, který se pak pomocí transpilace transformuje do známých objektů. Můžeme psát celé struktury prvků, aniž bychom se museli starat o to, jak budou převedeny do stromu DOM a přidány na stránku. O to se postará JSX a samotný React.[46]

---

[44] WIERUCH, Robin. React DOM. In: *The Road to React*. Leanpub, 2021, s. 34. ISBN 9781720043997. [cit. 2024-03-15].
[45] EDUCATIVE. *What is unidirectional data flow in React?* Online. Dostupné z: https://www.educative.io/answers/what-is-unidirectional-data-flow-in-react. [cit. 2024-03-15].
[46] FLANAGAN, David. JSX: Markup Expressions in JavaScript. In: *JavaScript: The definitive guide: Master the world's most-used programming language.*. Seventh Edition. O'Reilly. ISBN 978-1491952023. [cit. 2024-03-15].

Jak to funguje? Pomocí syntaxe JSX můžeme deklarovat proměnnou a přiřadit jí jednoduchý prvek JSX, který bude mít hlavičku h1:

```
6   const title = <h1> JSX </h1>
```

Obrázek 7

Při parsování našeho souboru JS si transpiler uvědomí, že proměnná title není jen řetězec, ale element JSX, a převede ji do podoby, která je pro React dále srozumitelná. Knihovna React pro něj zase zavolá speciální knihovní metodu: React.createElement(). V podstatě – ekvivalentní by bylo, kdybychom napsali:

```
13   const title = React.createElement('h1', {}, 'JSX');
```

Obrázek 8

Jinými slovy, vždy bychom měli mít na paměti, že nezapisujeme řetězce HTML, ale objekty obsahující informace o prvku React. Proto můžeme pro značky v JSX bezpečně používat libovolné platné názvy XML a předávat do nich libovolná data. React je dokonale zpracuje a použije při vykreslování do stromu DOM. Nakonec se z našeho h1 samozřejmě stane html element h1. Než ho uvidíme v prohlížeči, projde celou řadou rutin React.

Při práci s JSX byste si měli uvědomit, že nepracujeme se šablonovacím enginem, ale s plnohodnotným JavaScript se speciální syntaxí. Na rozdíl od běžného šablonovacího enginu, který může přijímat pouze řetězcové hodnoty, můžeme v JSX provádět libovolné výrazy JS uvnitř kudrlinkových závorek.

Stojí také za to vědět, že React neinteraguje přímo s JSX, ale bere již přepracované objekty Babel a pracuje s nimi. Samozřejmě můžeme napsat React.createElement() tak, že obejdeme JSX, ale v tomto případě je JSX "syntaktickým cukrem", který nám umožňuje zjednodušit práci s komponentami React. [47. 48.]

3.4.2.5 **Životní cyklus komponent**

Během svého provozu prochází komponenta několika fázemi životního cyklu. V každé z těchto fází je volána určitá funkce, v níž můžeme definovat některé akce:[49]

---

[47] WIERUCH, Robin. React JSX. In: *The Road to React*. Leanpub, 2021, s. 19. ISBN 9781720043997. [cit. 2024-03-15].
[48] REACT.JS. *Learn React*. Online. Writing Markup with JSX. Dostupné z: https://react.dev/learn/writing-markup-with-jsx. [cit. 2024-03-15].
[49] W3 SCHOOLS. *React Lifecycle*. Online. Dostupné z: https://www.w3schools.com/react/react_lifecycle.asp. [cit. 2024-03-15].

1. constructor(props): konstruktor, ve kterém se provede počáteční inicializace komponenty
2. static getDerivedStateFromProps(props, state): volá se těsně před vykreslením komponenty. Tato metoda nepřistupuje k aktuálnímu objektu komponenty (tj. přistupuje k objektu komponenty prostřednictvím této metody) a měla by vrátit objekt pro aktualizaci objektu stavu nebo hodnotu null, pokud není co aktualizovat.
3. render(): vykreslí komponentu
4. componentDidMount(): volá se po vykreslení komponenty. Zde lze provádět požadavky na vzdálené zdroje
5. componentWillUnmount(): volá se před odstraněním komponenty z DOM

Kromě těchto hlavních fází nebo událostí životního cyklu existuje také řada dalších funkcí, které jsou volány při aktualizaci stavu po počátečním vykreslení komponenty, pokud dojde k aktualizacím komponenty:[50]

1. static getDerivedStateFromProps(props, state)
2. shouldComponentUpdate(nextProps, nextState): volá se při každé aktualizaci objektu props nebo state. Nový objekt props a stav jsou předány jako parametr. Tato funkce by měla vrátit true (měla by provést aktualizaci) nebo false (ignorovat aktualizaci). Ve výchozím nastavení se vrací true. Pokud však funkce vrátí false, zakážeme aktualizaci komponenty a následné funkce nebudou spuštěny.
3. render(): vykreslí komponentu (pokud funkce shouldComponentUpdate vrátí true)
4. getSnapshotBeforeUpdate(prevProps, prevState): volá se těsně před aktualizací komponenty. Umožňuje komponentě získat informace z DOM před případnou aktualizací. Vrací nějaký individuální aspekt jako hodnotu, která je předána jako třetí parametr metodě componentDidUpdate() a může být zohledněna v komponentě componentDidUpdate při její aktualizaci. Pokud není co vracet, vrátí se hodnota null.
5. componentDidUpdate(prevProps, prevState, snapshot): volá se ihned po aktualizaci komponenty (pokud shouldComponentUpdate vrátí true). Jako parametry jsou

[50] W3 SCHOOLS. *React Lifecycle*. Online. Dostupné z: https://www.w3schools.com/react/react_lifecycle.asp. [cit. 2024-03-15].

předány staré hodnoty objektů props a state. Třetím parametrem je hodnota vrácená metodou getSnapshotBeforeUpdate.[51, 52]

3.4.3 React Hooks

Hooky jsou funkce, které můžete použít k "napojení" na stav životního cyklu React a metody funkčních komponent. Hooky nefungují uvnitř tříd – místo toho umožňují používat React bez tříd.

K čemu slouží?

- Hooky usnadňují opakované použití stavové logiky v různých komponentách.
- Hooky umožňují rozdělit jednu komponentu na malé funkce podle jejich účelu (např. odběr nebo načítání dat), nikoli podle metod životního cyklu.
- Zjednodušují organizaci kódu a umožňují jeho opakované použití.
- Eliminují potřebu porozumět tomu, jak to funguje v jazyce JavaScript, jehož chování se liší od většiny jazyků.
- Hooky poskytují nový výkonný způsob opakovaného použití kódu v komponentách.

Podívejme se na některé z nejpopulárnějších hooků

Hook useState je jednoduchý hook pro vývojáře, ale důležitý pro celou aplikaci. Začněme nejjednodušším a nejdůležitějším hookem – useState. Už ze samotného názvu je zřejmé, že souvisí se stavem komponenty. Právě díky němu mají funkční komponenty nějaký stav. Uvedeme si jednoduchý příklad použití tohoto hooku a poté si jej podrobně rozebereme.[53]

---

[51] REACT.JS. *Learn React*. Online. Synchronizing with Effects. Dostupné z: https://react.dev/learn/synchronizing-with-effects. [cit. 2024-03-15].
[52] REACT.JS. *Learn React*. Online. State: A Component's Memory. Dostupné z: https://react.dev/learn/state-a-components-memory. [cit. 2024-03-15].
[53] REACT.JS. *Hooks API Reference*. Online. Dostupné z: https://legacy.reactjs.org/docs/hooks-reference.html#usestate. [cit. 2024-03-15].

```
1  const App = () => {
2      const [value, valueChange] = useState(0);
3
4      return (
5        <div>
6          {value}
7          <button onClick={() => valueChange(value + 1)}>
8            Zvyseni hodnoty o 1
9          </button>
10       </div>
11     );
12   };
```

Obrázek 9 useState

Vytvoří stav a metodu, která tuto hodnotu změní. Hook useState v podstatě přijímá jako parametr počáteční hodnotu, což znamená, že na začátku bude mít naše hodnota hodnotu 1. A vrací useState pole dvou prvků: první je stav, druhý je metoda, která jej změní. Vývojáři hooků použili docela šikovný přístup. Pomocí destrukce umožňují nastavit libovolnou hodnotu stavu a metody s minimálním množstvím kódu.

Věnujte pozornost řádkům 7-9. Zde jsme přidali obslužný program pro událost stisknutí tlačítka. Vysvětlím: po kliknutí na tlačítko zavoláme metodu valueChange a pošleme tam novou hodnotu – v našem případě zvýšenou o jedničku.

Jinak je vše stejné jako při normálním stavu komponenty. Hlavní rozdíl: v komponentě třídy můžeme vytvořit pouze jeden běžný stav komponenty, zatímco ve funkční komponentě jich můžeme vytvořit několik a budou na sobě nezávislé, ale každý z nich způsobí vykreslení komponenty. [54]

Hooky useEffect a useLayoutEffect slouží k oživení komponent, přesněji řečeno k oživení metod. Používají se k provádění některých operací v různých fázích životnosti součásti. K tomuto účelu slouží dva hooky – useEffect a useLayoutEffect. Jsou si podobné až na drobný rozdíl ve vykreslování. V případě useLayoutEffect nezačne React vykreslovat sestavený strom DOM, dokud není spuštěn useLayoutEffect. Vezmeme-li useEffect, začne React okamžitě vykreslovat sestavený DOM, aniž by čekal na spuštění useEffect.[55]

Pomocí těchto dvou hooků ve funkčních komponentách můžeme simulovat práci třech metod životního cyklu – componentDidMount, componentDidUpdate, componentWillUnmount. Přesněji řečeno jejich práci simuluje useLayoutEffect, protože ve

---

[54] REACT.JS. *Learn React*. Online. UseState. Dostupné z: https://react.dev/reference/react/useState. [cit. 2024-03-15].
[55] REACT.JS. *Hooks API Reference*. Online. Dostupné z: https://legacy.reactjs.org/docs/hooks-reference.html#usestate. [cit. 2024-03-15].

třídě komponent se vykreslování DOM-stromu spouští až po provedení metody componentDidMount. [56][57]

Hook useContext. Chceme-li komponentě předat nějaká data, můžeme použít props. Existuje však i alternativní způsob – kontext. Kontext umožňuje předávat data z rodičovské komponenty do podřízené komponenty, přičemž se obejdou mezičlánky.

Vytvořil jsem malý příklad, který vám umožní pochopit, jak to funguje. Máme tři komponenty. První z nich je "External", druhá je „Intermediate", tedy mezilehlá, a třetí se bude jmenovat „Internal". Ve skutečnosti budou všechny vnořené jedna do druhé. Naším úkolem je přenést data z komponenty External do komponenty Internal, přičemž komponentu Intermediate obejdeme, protože s ní tato data nemají nic společného.[58]

```
import {createContext, useContext} from "react";

const MyContext = createContext("without provider");

const External = () => {
  return (
    <MyContext.Provider value="Hello, i am External">
      <Intermediate />
    </MyContext.Provider>
  );
};

const Intermediate = () => {
  return <Internal />;
};

const Internal = () => {
  const context = useContext(MyContext);

  return `I am Internal component. I have got the message from External: "${context}"`;
};
```

Obrázek 10 useContext

Chceme-li použít kontext, vytvoříme objekt MyContext voláním metody createContext. V komponentě External obalíme komponentu Intermediate komponentou MyContext.Provider. Tím říkáme, že všechny vnořené komponenty budou moci přistupovat k předávaným datům tím, že je vloží do parametru value.

A budou k dispozici pouze v těch komponentách, ve kterých je potřebujeme. K tomu musíme použít háček useContext, který bude mít jako argument objekt MyContext. Háček useContext vrátí data předaná do parametru value komponenty MyContext.Provider, která

---

[56] REACT.JS. *Learn React*. Online. UseEffect. Dostupné z: https://react.dev/reference/react/useEffect. [cit. 2024-03-15].
[57] REACT.JS. *Learn React*. Online. UseLayoutEffect. Dostupné z: https://react.dev/reference/react/useLayoutEffect. [cit. 2024-03-15].
[58] REACT.JS. *Hooks API Reference*. Online. Dostupné z: https://legacy.reactjs.org/docs/hooks-reference.html#usestate. [cit. 2024-03-15].

vložíme do proměnné context. Všimněte si, že jsme jako argument příkazu createContext předali řetězec ("bez kontextu"). Jeho hodnota přejde do kontextové proměnné pro případ, že byste náhle zapomněli vytvořit obal MyContext.Provider, tj. pomůže vám, abyste se z nepozornosti nespletli.[59]

Díky háčku useContext můžete kontext používat ve funkčních komponentách a data se dostanou jen do těch komponent, kde jsou potřeba. Zbavíte se tak také problému s vrtáním kapek. [60]

[59] REACT.JS. *Hooks API Reference*. Online. Dostupné z: https://legacy.reactjs.org/docs/hooks-reference.html#usestate. [cit. 2024-03-15].
[60] REACT.JS. *Learn React*. Online. UseContext. Dostupné z: https://react.dev/reference/react/useContext. [cit. 2024-03-15].

4 Vlastní práce

4.1 Popis

Před sestavením logického návrhu webu se musíte rozhodnout pro klíčové uživatele, stručně popsat rozvržení a sestavit funkčnost.

4.1.1 Cílová skupina

Pro určení cílové skupiny se uchýlím k metodě 5W. Název metody je odvozen od prvních písmen pěti otázek, na které je třeba podle zakladatele metody Sherringtona odpovědět:

- Co? (When?) Jaký druh výrobku se prodává, o jaký druh výrobku se jedná.
- Kdo? (Who?) Kdo by mohl být potenciálním kupujícím, jak vypadá.
- Proč? (Why?) Proč bude kupující preferovat právě tento výrobek nebo službu, co ho motivuje ke koupi.
- Kdy? (When?) Kdy kupující pravděpodobně provede nákup, za jakých okolností a podmínek může k nákupu dojít.
- Kde? (Where?) Kde bude osoba nákup provádět. Jaké místo bude pro tento proces optimální.

Odpovědi na tyto otázky vám pomohou lépe pochopit nejen to, komu můžete své produkty nabídnout, ale také kde a za jakých podmínek je to nejvhodnější.[61]

Tímto způsobem získáme odpovědi:

1. Oblečení a tenisky.
2. Potenciálního klienta vidím jako osobu ve věku 20-45 let s průměrným měsíčním příjmem 40 000 korun.
3. Protože klient chce vypadat stylově, módně.
4. Přibližně jednou měsíčně pro aktualizaci šatníku nebo několikrát do roka jako dárek.
5. V internetovém obchodě.

[61] GALIANA, David. The 5W1H Method : Project Management defined and applied. Online. *Wimi*. S. 1. Dostupné z: https://www.wimi-teamwork.com/blog/the-5w1h-method-project-management-defined-and-applied/. [cit. 2024-03-15].

4.1.2 **Funkčnost**

Při vytváření funkčnosti webu budu vycházet z principu tvorby webových stránek, který navrhl odborník na UX Jess Garrett.

UX, tedy user experience, se z angličtiny překládá jako "uživatelská zkušenost" nebo "interakční zkušenost". UX spojuje vše, co lze realizovat prostřednictvím funkčnosti a designu, aby se uživatel dostal k požadovanému cíli. UX spočívá v tom, že interakce s webovými stránkami je pro uživatele pohodlná a příjemná. Tak, aby návštěvník snadno dosáhl logického bodu a provedl užitečnou akci, kvůli které se na vás obrátil: získal potřebné informace, zjistil harmonogram společnosti, cenu produktu nebo se přihlásil k odběru. Tato užitečná akce zase uživatele dovede k cílové akci – nákupu.

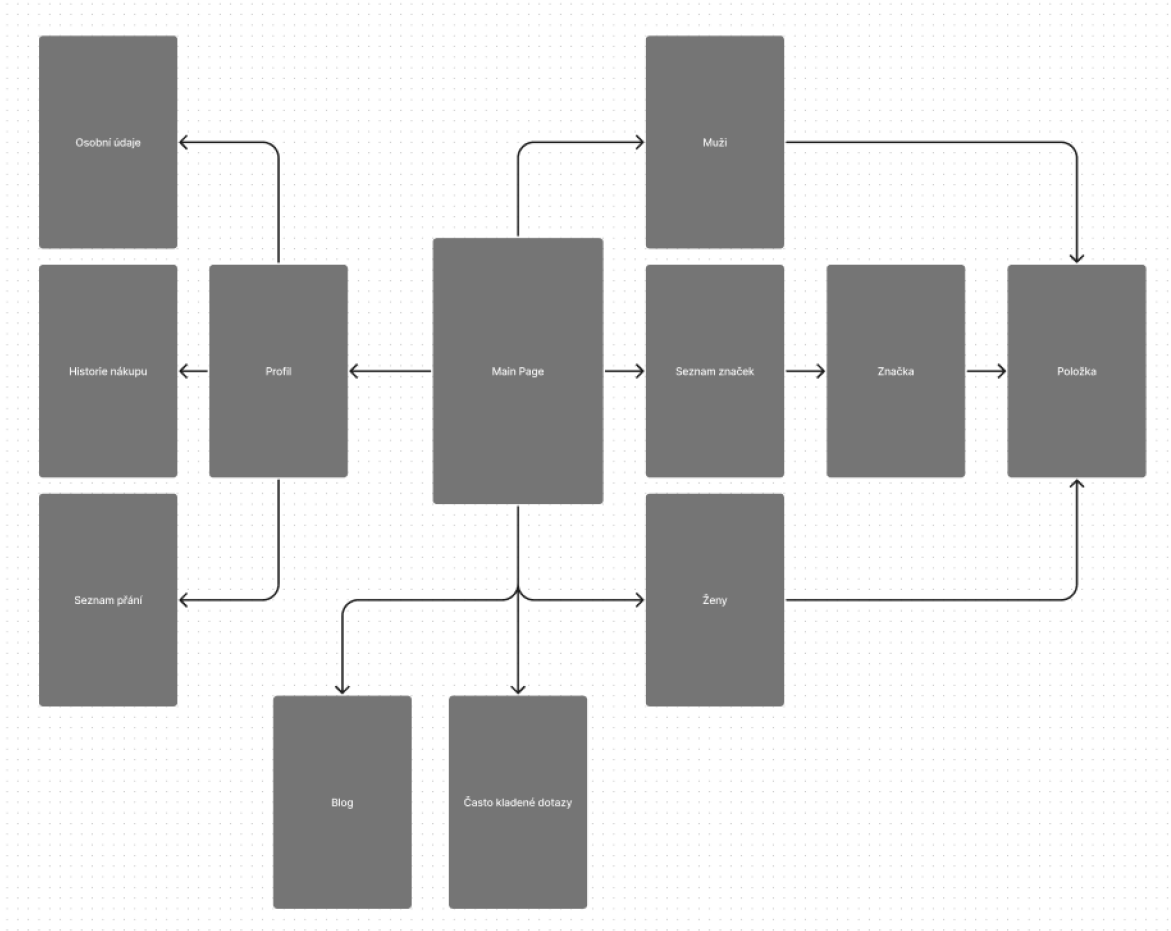
Jess Garrett navrhuje rozdělit práci na funkčním obsahu webu do pěti úrovní:

- Strategie – popište cíle a záměry webu, prostudujte CA.
- Schopnosti – zjistěte, jaké funkce pomáhají řešit problémy uživatele.
- Struktura – navrhněte architekturu webu v závislosti na zkušenostech uživatele.
- Rozvržení – připravte prototyp webu.
- Povrch – navrhněte design webu.[62]

Web by tedy měly především fungovat rychle, prvky navigačního panelu by měly být intuitivní. Struktura webu by měla obsahovat navigační panel, seznam věcí, seznam značek, stránku s dotazy, blog s módními novinkami, osobní účet, nákupní košík a oblíbené položky. Než však přejdeme k logickému návrhu, je třeba sestavit User Flow, mapu webu a uživatelské scénáře.

[62] GARRETT, Jesse James. *The Elements of User Experience: User-Centered Design for the Web and Beyond*. 2nd. 2010. ISBN 978-0321683687. [cit. 2024-03-15].

4.1.3 Mapa webu



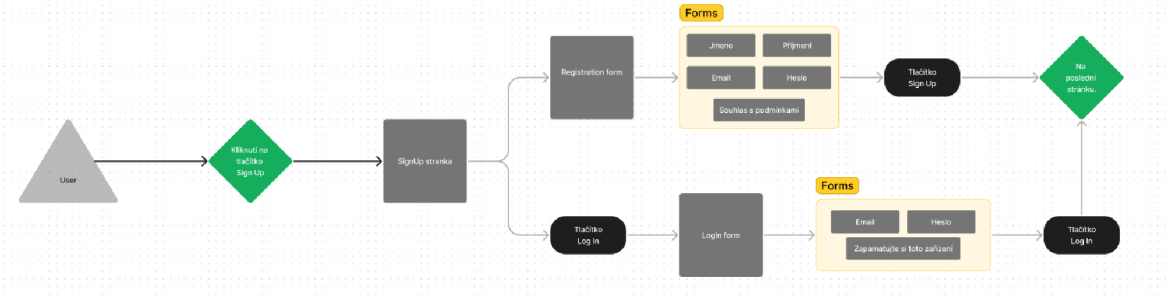
Obrázek 11 Mapa webu

4.1.4 User flow

Uživatelské toky, známé také jako „User flow", „UX flow", „user flow charts", „interaction flows", „activity flows", „UI flows", „navigation flows", „user flow diagrams", „task flow diagrams" nebo vývojové diagramy. Jsou to diagramy, které ukazují kompletní cestu, kterou uživatel při používání produktu prochází. Uživatelský tok popisuje postup uživatele produktem a ukazuje každý krok, který uživatel učiní – od bodu vstupu (začátku) až po konečnou interakci (dosažený smysluplný cíl, získaná hodnota).[63]

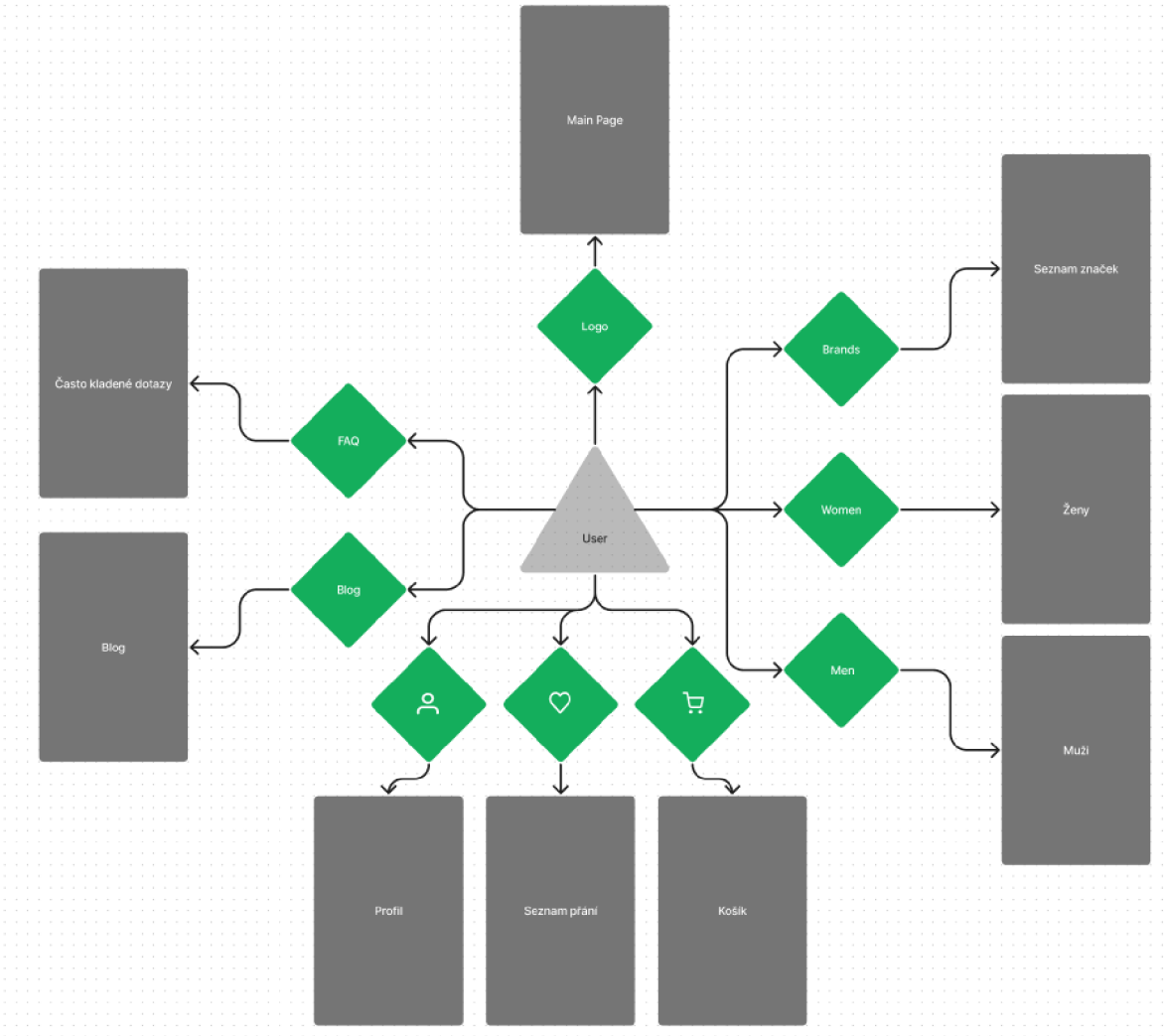
[63] INTERACTION DESIGN FOUNDATION. *User Flows*. Online. Dostupné z: https://www.interaction-design.org/literature/topics/user-flows. [cit. 2024-03-15].

4.1.4.1 Registrace / Přihlášení



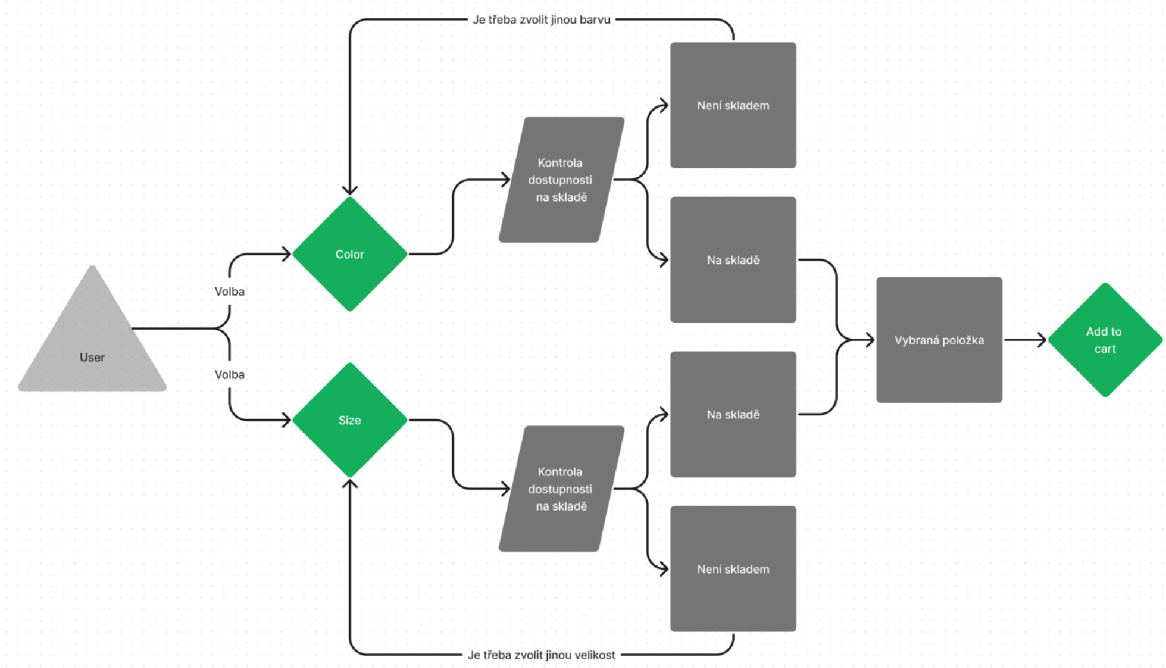
Obrázek 12 Registrace

4.1.4.2 Navigační panel



Obrázek 13 Navigační panel

4.1.4.3 **Přidání položky do košíku**



Obrázek 14 Přidání položky do košíku

4.1.5 **Uživatelské scénáře**

Uživatelské scénáře jsou vizuální příběhy chování uživatelů založené na jedné osobě, která představuje jednoho z uživatelů webových stránek nebo aplikace. Každý scénář je založen na konkrétním cíli, kterého má být dosaženo. Takové scénáře se používají ve výzkumu UX, aby nám pomohly vizualizovat kroky, které uživatel podniká k dosažení konkrétního cíle, např. koupě produktu, vyhledání informací nebo použití služby na webové stránce, v mobilní aplikaci. US popisuje roli uživatele v produktu, jeho potřebu a výsledek, kterého dosáhne, pokud k události dojde.[64]

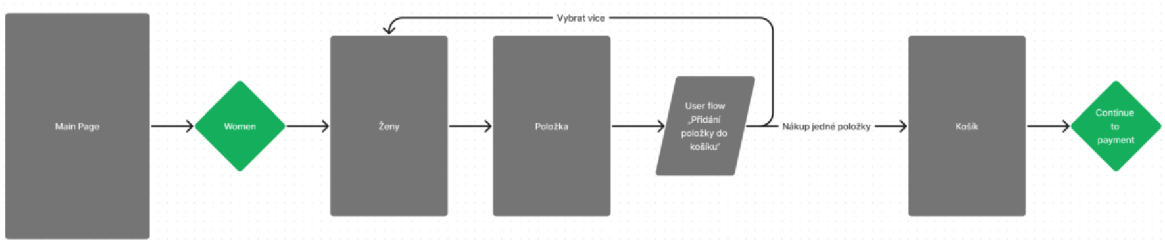
4.1.5.1 **Nákup**

Osoba: Jeanne, 27letá studentka s neformálním vkusem, nakupuje raději online, aby ušetřila čas a získala větší výběr.

Cíl: Koupit jedinečný dárek pro kamarádku za přijatelnou cenu.

---

[64] INTERACTION DESIGN FOUNDATION. *User Scenarios*. Online. Dostupné z: https://www.interaction-design.org/literature/topics/user-scenarios. [cit. 2024-03-15].

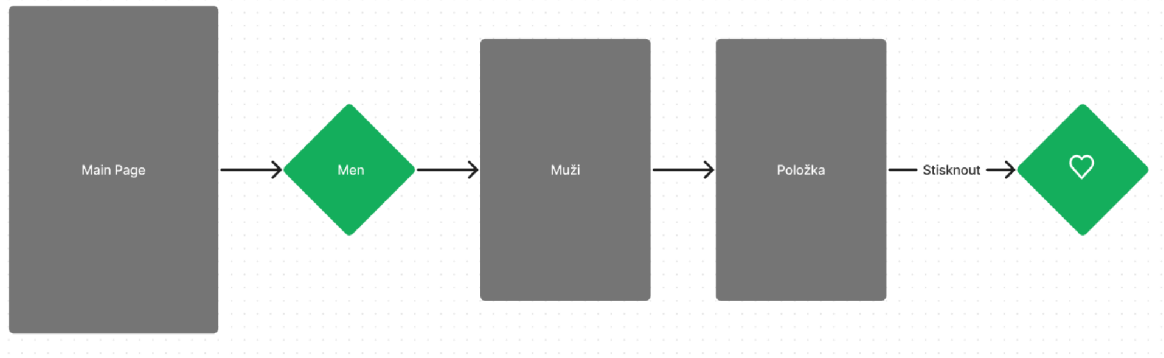


Obrázek 15 User scenario 1

4.1.5.2 **Přidání položky do seznamu přání**

Osoba: Pavel, 25 let, kancelářský pracovník, rád se obléká módně.

Cíl: Vytvořit seznam přání.

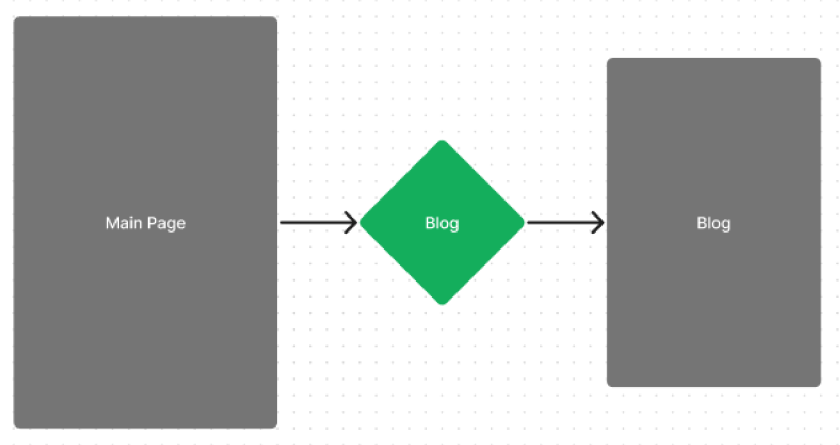


Obrázek 16 User scenario 2

4.1.5.3 **Prohlížení módních novinek**

Osoba: James, 46 let, módní návrhář. Veškerý svůj volný čas věnuje sledování módních novinek, nových návrhů a módních přehlídek.

Cíl: Číst novinky.

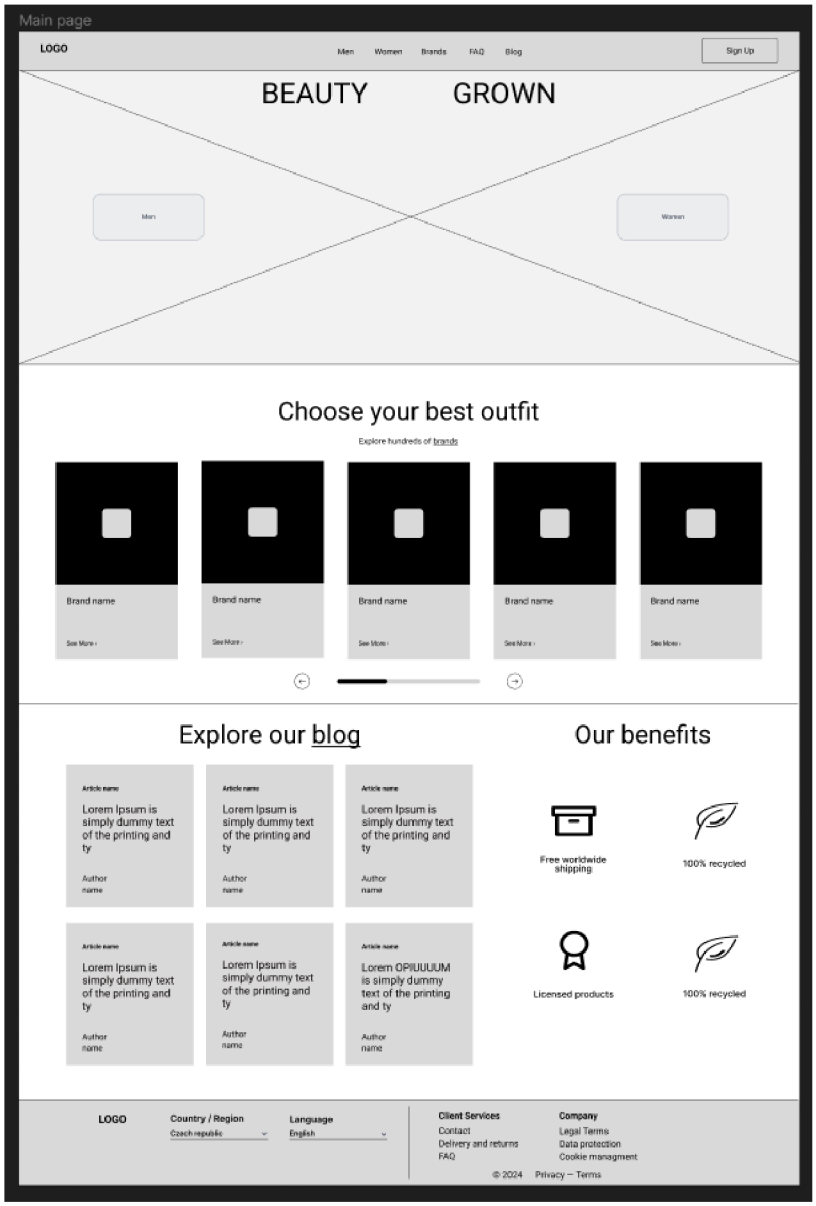


Obrázek 17 User scenario 3

4.2 Logický design

Po vytvoření logického návrhu jsme získali 16 rozvržení, z nichž 2 jsou přihlášení a registrace a 1 je rozvržení filtru. „“

4.2.1 Hlavni stránka



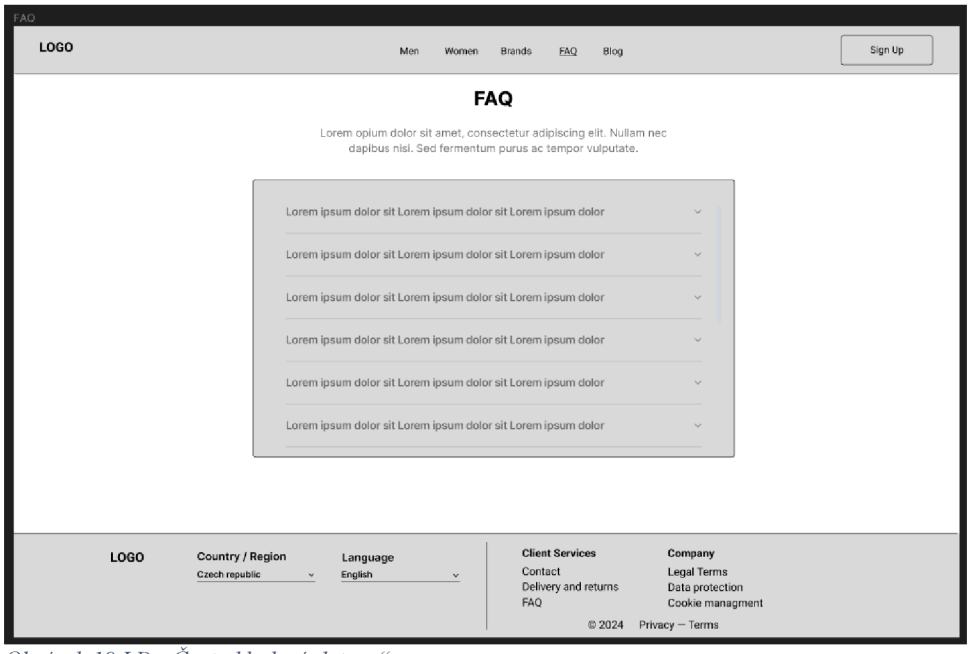
Obrázek 18 LD „Hlavní stránka”

Hlavní stránka se skládá z 5 prvků:

1. Navigační panel. Skládá se ze 3 velkých prvků. Logo je umístěno vlevo, odkazy uprostřed, registrační tlačítko vpravo. Prvky navigačního panelu budou na všech stránkách stejné, změní se pouze registrační tlačítko.

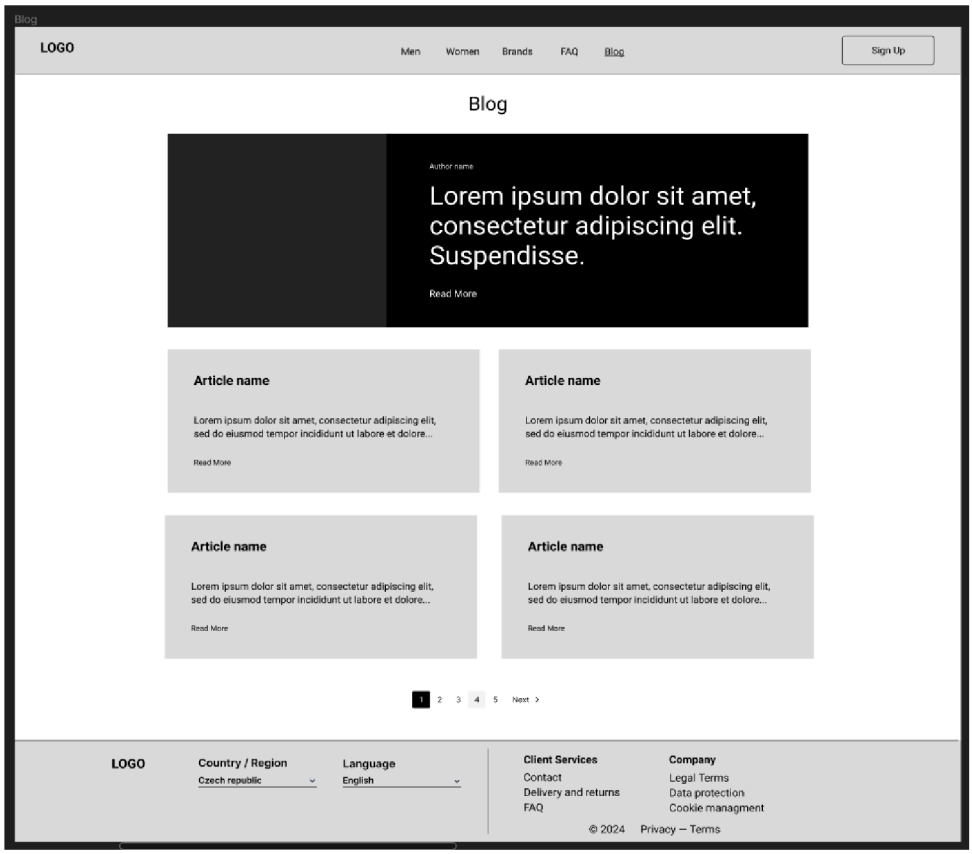
2. Obrázek. Na obrázku vlevo bude muž a tlačítko umístěné vlevo povede na seznam pánského oblečení. Na pravé straně bude žena s příslušným tlačítkem.
3. Seznam značek.
4. Blog s nejnovějšími články a Výhody používání tohoto internetového obchodu.
5. Zápatí. Včetně jazyka stránky, země a různých právních dokumentů.

4.2.2 Často kladené dotazy, Blog a Client services / Company



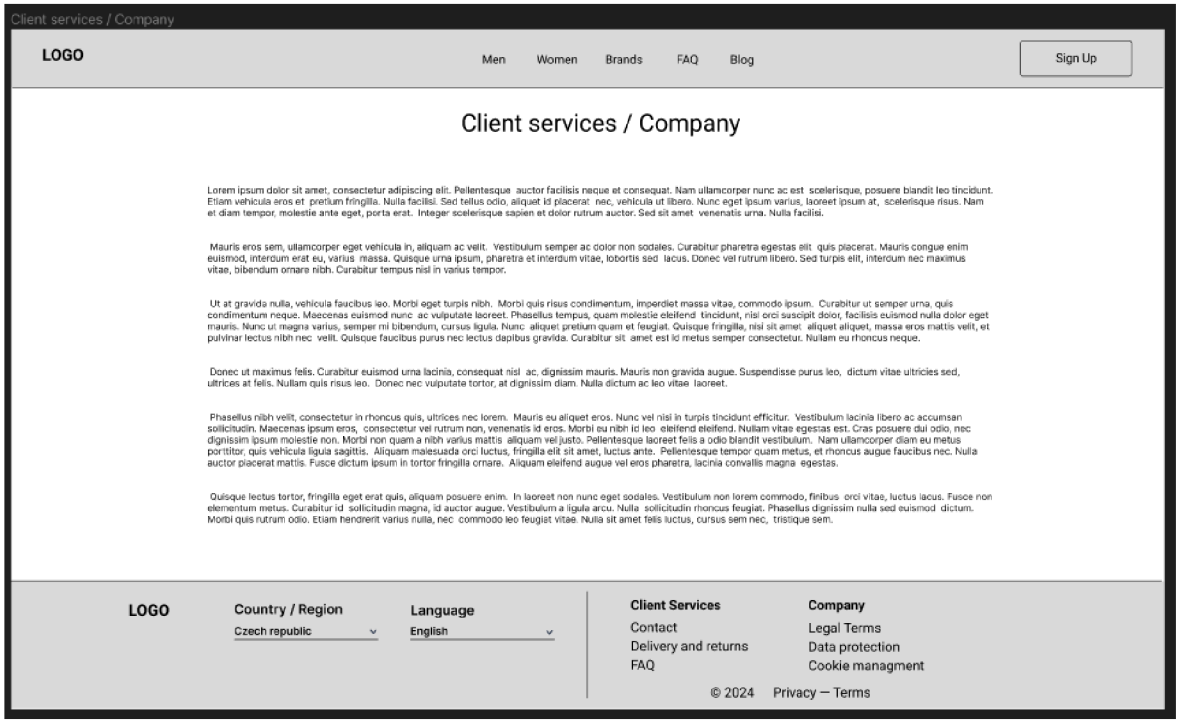
Obrázek 19 LD „Často kladené dotazy“

Stránka s otázkami obsahuje název stránky, stručný popis a seznam otázek, na které se po kliknutí zobrazí odpověď.



Obrázek 20 LD „Blog“

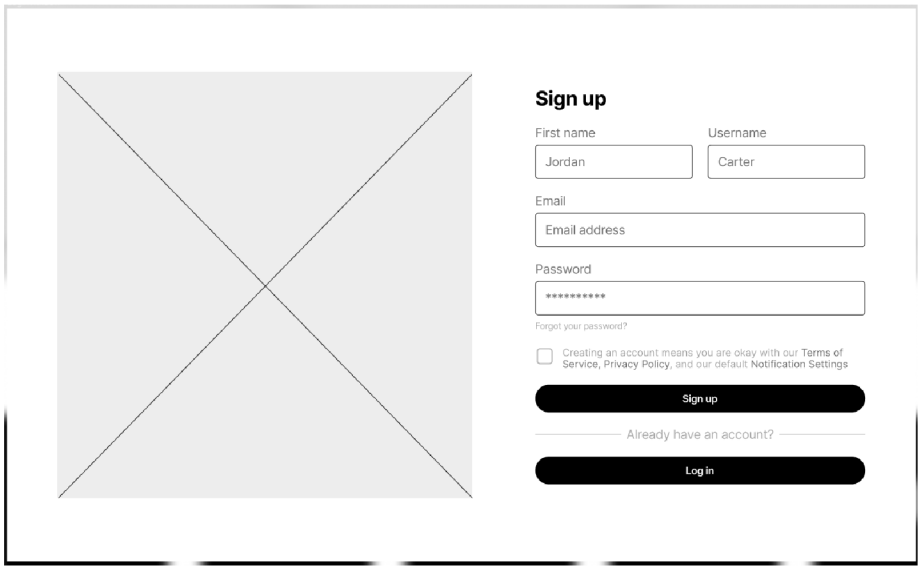
Na stránce blogu si budete moci přečíst nejnovější zprávy a oznámení. Nejdůležitější zprávy/aktuality týdne budou zvýrazněny černou barvou a velikostí.



Obrázek 21 LD „Client service / Company“

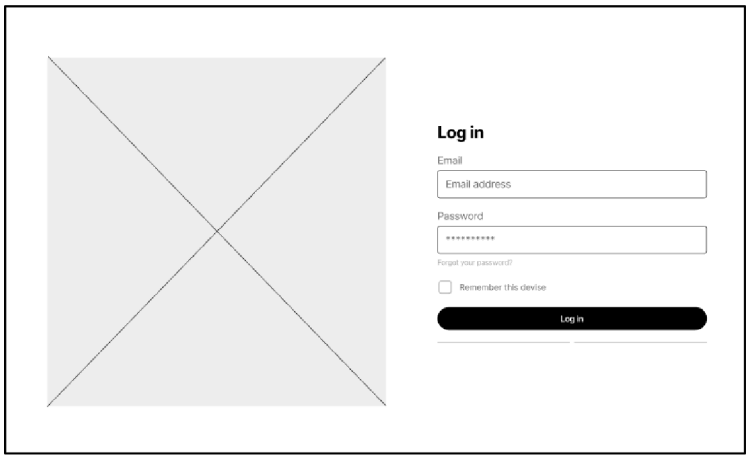
Obecná šablona stránek: „Contact“, „Delivery and returns“, „Legal Terms“, „Data Protection“, „Cookie management“, „©2024 Privacy – Terms”. Uprostřed stránky se zobrazí odpovídající text.

4.2.3 **Registrace / Přihlášeni**



Obrázek 22 LD „Registrace“

Po kliknutí na tlačítko „Sign up“ se v pravém horním rohu se uprostřed stranky zobrazí registrační pole, ve kterém musí zákazník vyplnit formulář. Pokud kliknete na tlačítko „Log in“ okno se rovněž změní na:

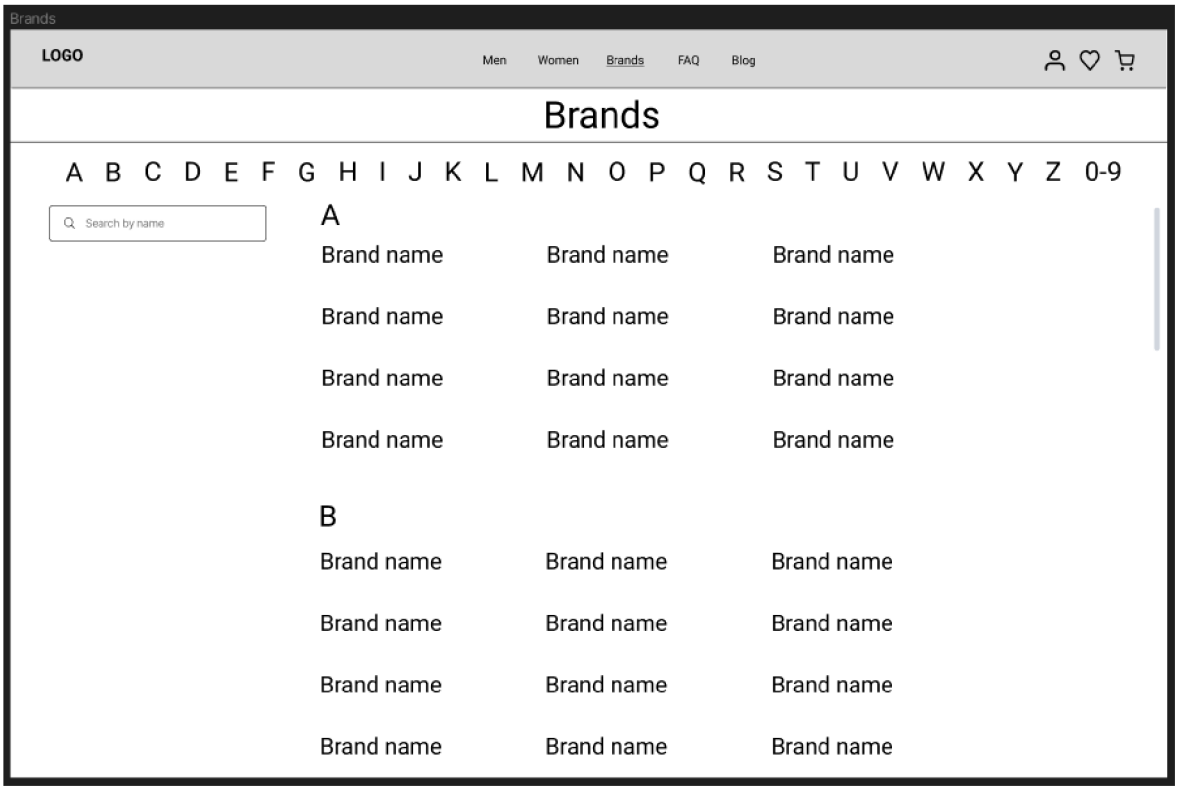


Obrázek 23 LD „Přihlášeni“

Obrázek 24 Ikonky

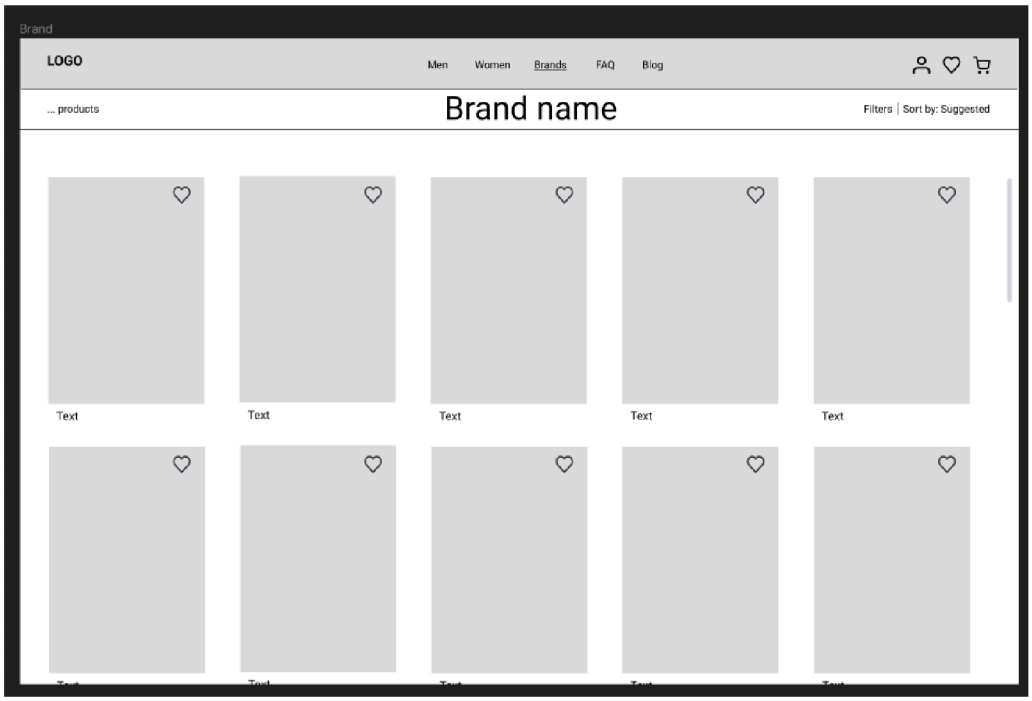
Při přihlášení k účtu se tlačítko v pravém horním rohu změní na 3 ikony. Jsou to ikony osobní skříňky, nákupního košíku a seznamu oblíbených produktů.

4.2.4 **Muži, Ženy, Značky**



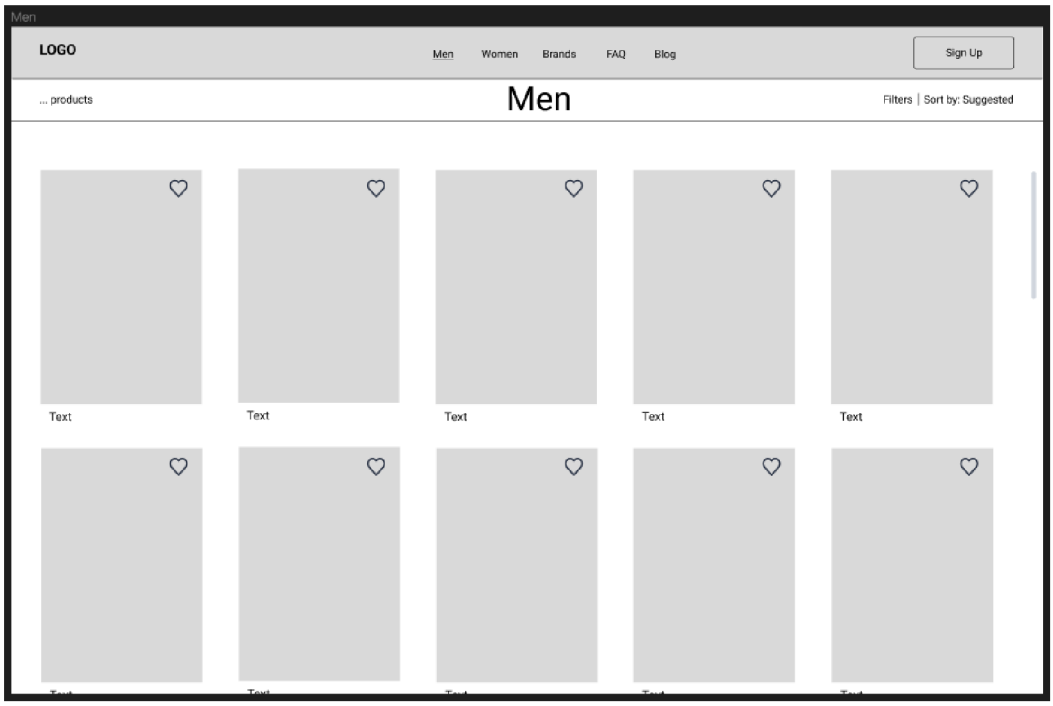
Obrázek 25 LD „Seznam značek“

Stránka značek bude obsahovat všechny značky seřazené podle abecedy. V horní části bude filtr v podobě písmen, který seřadí značky podle vybraného písmene. Po kliknutí na vybranou značku dojde k přesměrování na stránku vybrané značky, kde se nachází položky této značky.

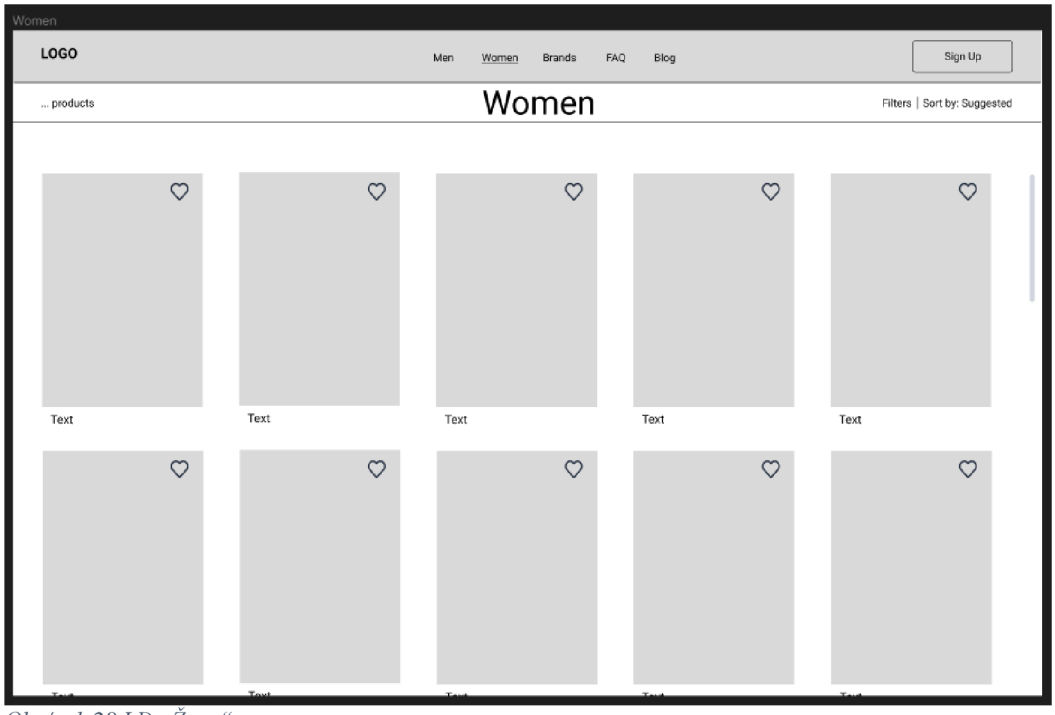


Obrázek 26 LD „Značka“

Na stránce vybrané značky najdete pouze produkty této značky. V levém horním rohu je napsán počet produktů prezentovaných na stránce, na pravé straně jsou filtry, které upřesňují vyhledávání, a také řazení podle nabídky, ceny. Stejná struktura bude platit i na stránkách: „Men“, „Women“.

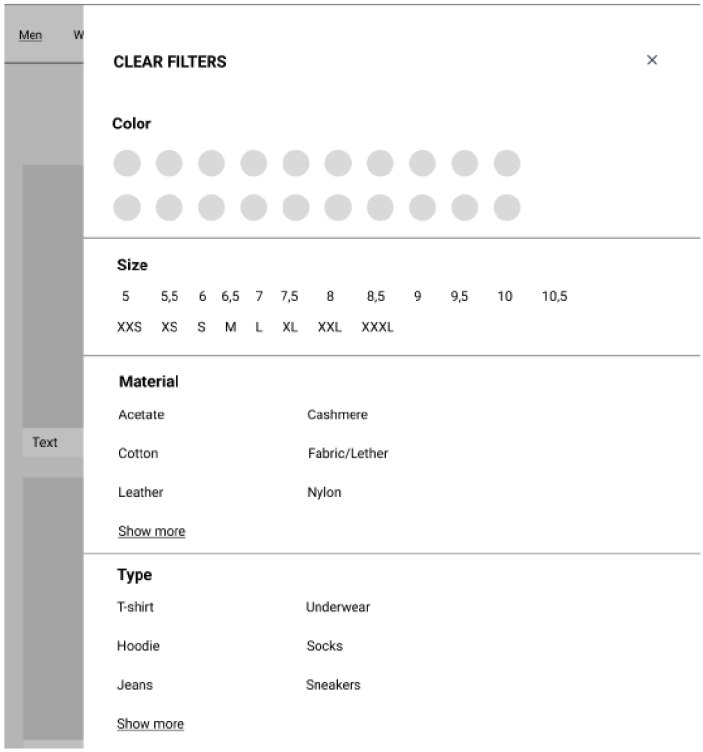


Obrázek 27 LD „Muži“



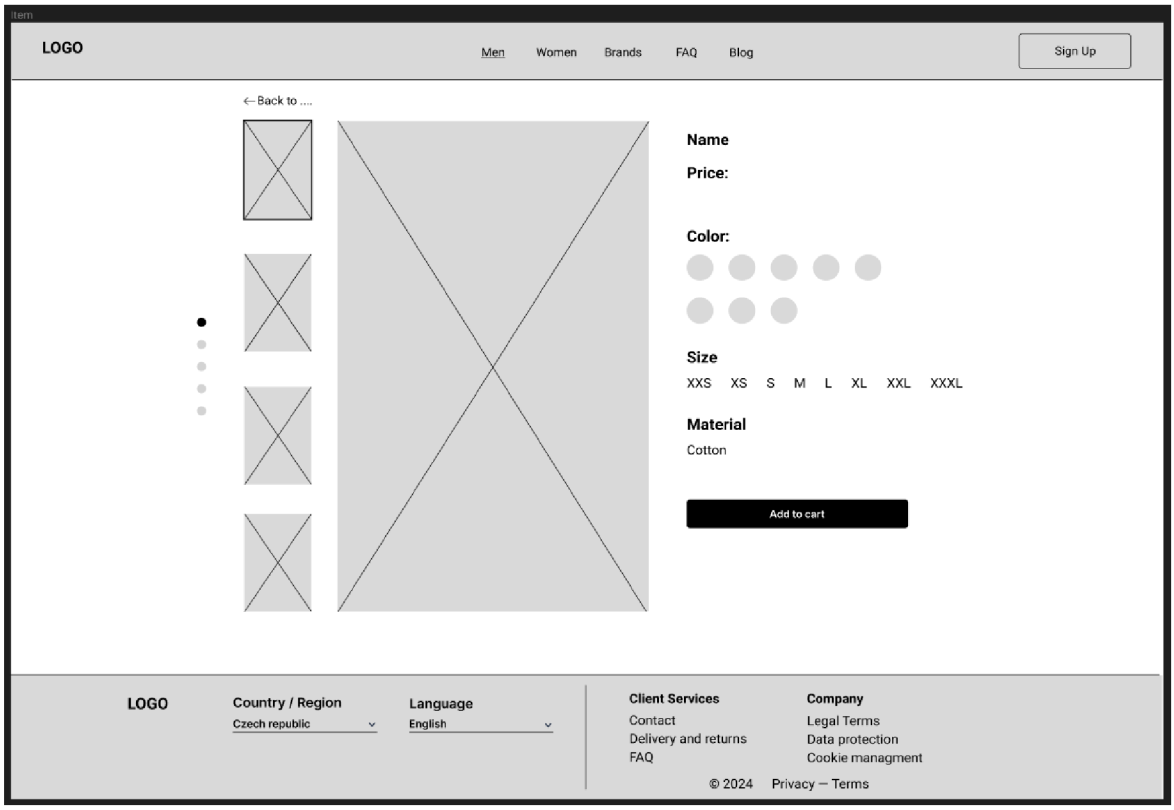
Obrázek 28 LD „Ženy“

Na stránkách, kde bude prezentováno oblečení, bude také možné třídit oblečení a tenisky pomocí filtrů. Při třídění můžete vybrat barvu, velikost, materiál a typ.



Obrázek 29 LD „Filtry“

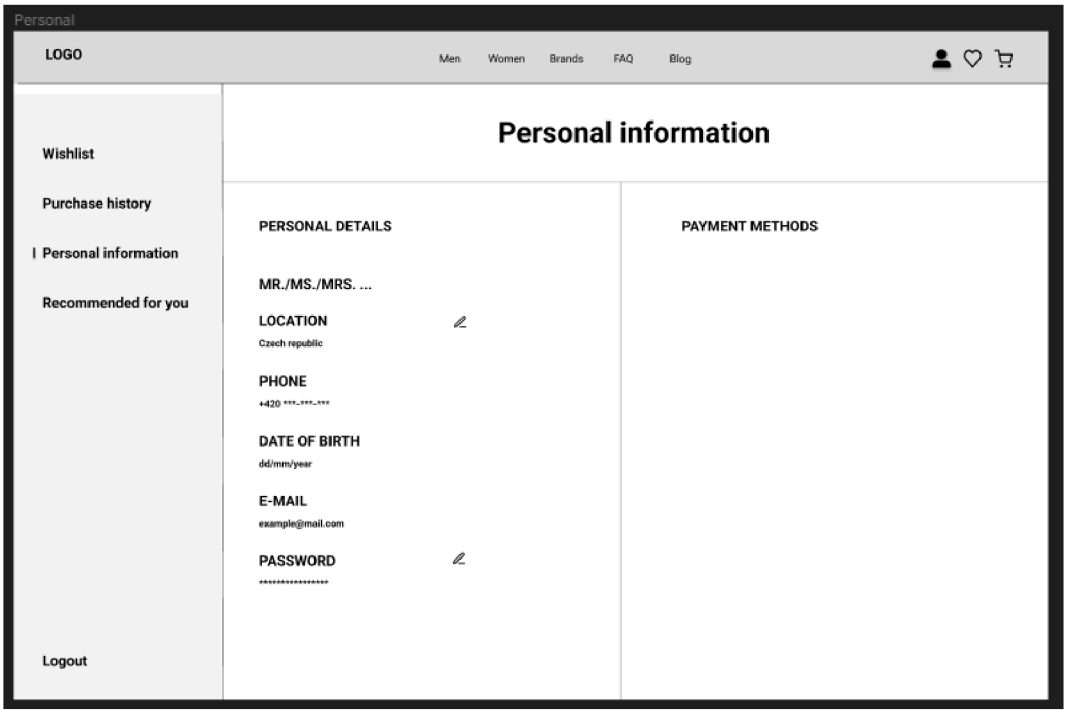
4.2.5 **Položka**



Obrázek 30 LD „Položka“

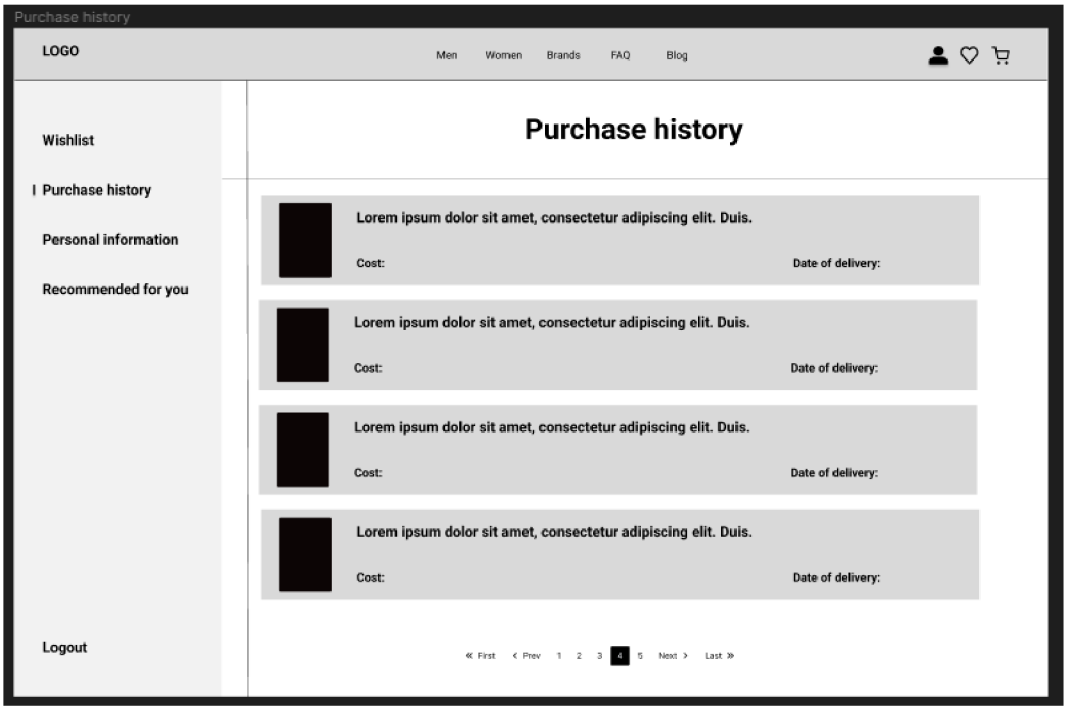
Na stránce produktu se zobrazí koláž fotografií z různých úhlů, název, cena, výběr barvy, mřížka velikostí, materiál, ze kterého je produkt vyroben, a také tlačítko pro přidání do košíku.

4.2.6 **Osobní údaje**

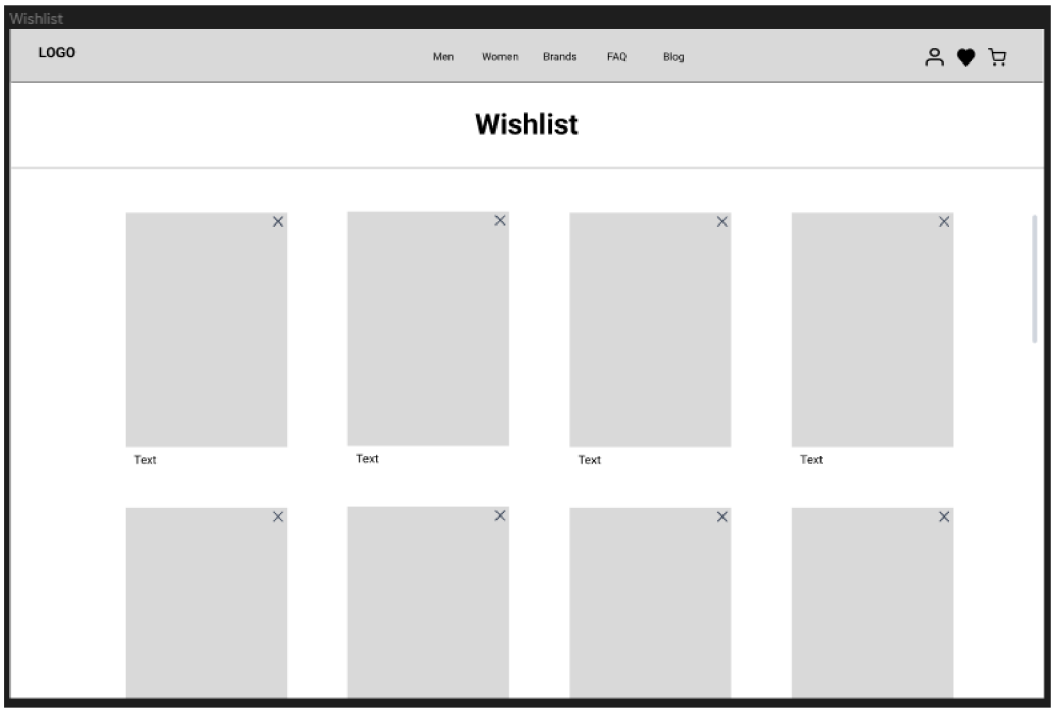


Obrázek 31 LD „Osobní údaje“

Osobní účet obsahuje všechny důležité informace o účtu, včetně propojených platebních metod. Kromě toho si můžete zobrazit historii nákupů, doporučení z obchodu a oblíbené produkty.

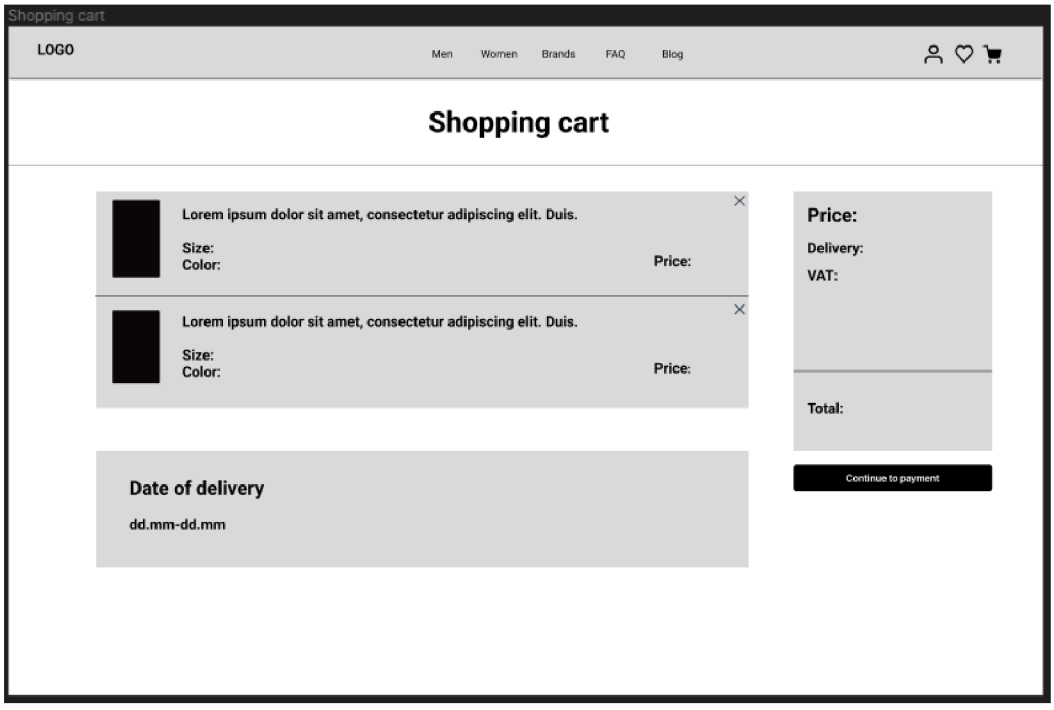


Obrázek 32 LD „Historie nákupu“



Obrázek 33 LD „Seznam přání“

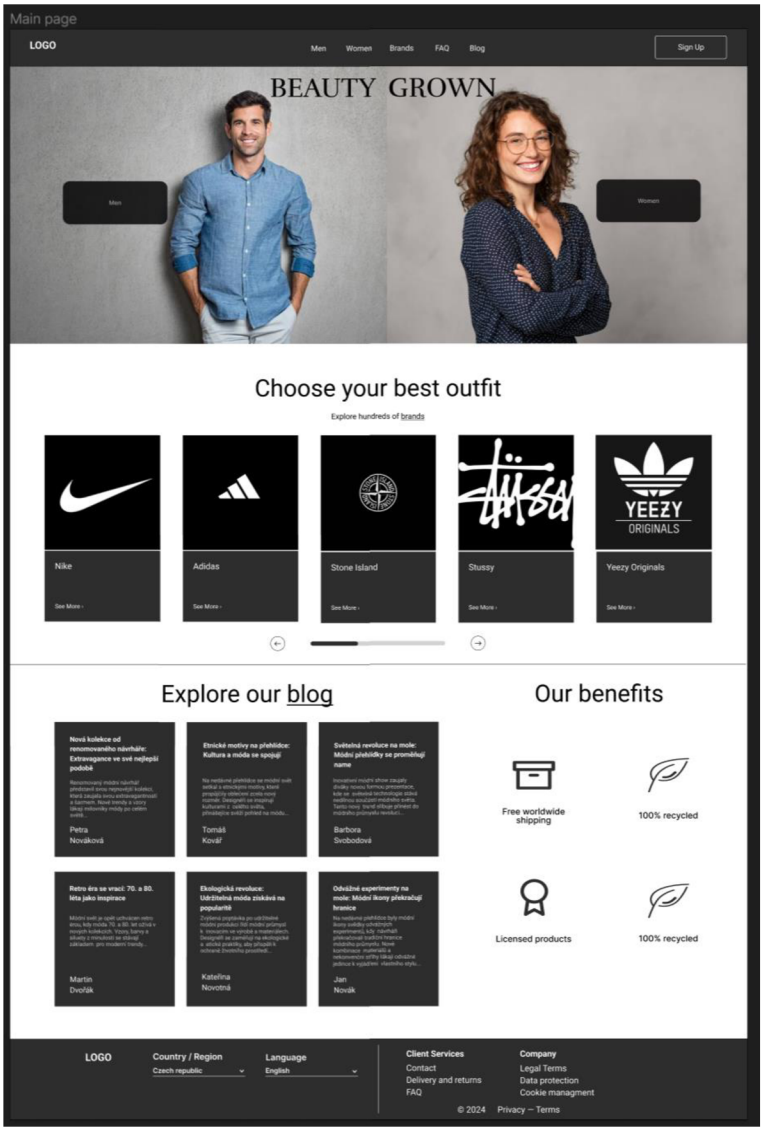
4.2.7 Nákupní košík



Obrázek 34 LD „Košík“

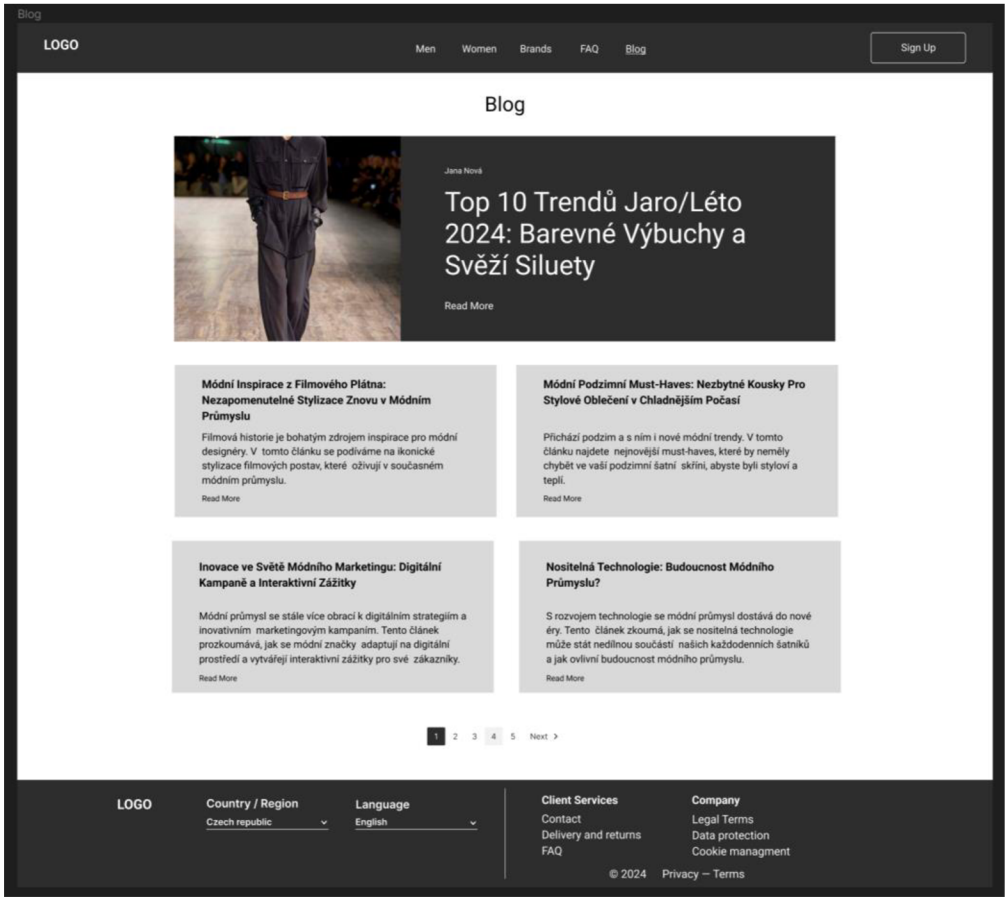
Na stránce košíku se zákazníkovi zobrazí 3 prvky, z nichž jeden obsahuje přidané zboží, druhý obsahuje informace týkající se přibližných termínů dodání a třetí obsahuje cenu zboží, dodání, daň a celkovou cenu.

4.3 Grafický design



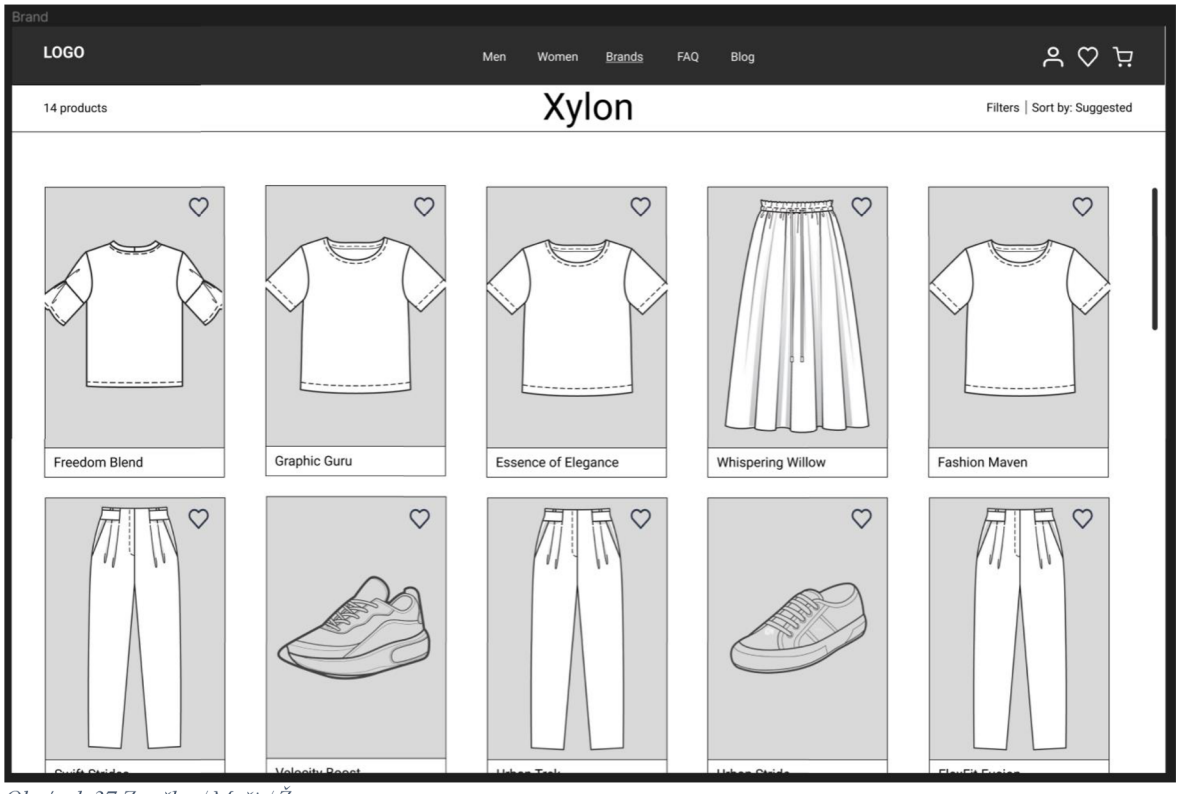
Obrázek 35 Hlavní stránka

Při tvorbě grafického stylu byl použit barevný minimalismus, kdy nejdůležitější části mají tmavou barvu a kontrastují s bílým pozadím, navigační panel i všechny hlavní prvky jsou vytvořeny v černých barvách, barva prvků na něm je pro kontrast bílá. Kromě toho jsou všechny aktivní odkazy v panelu označeny podtržením.



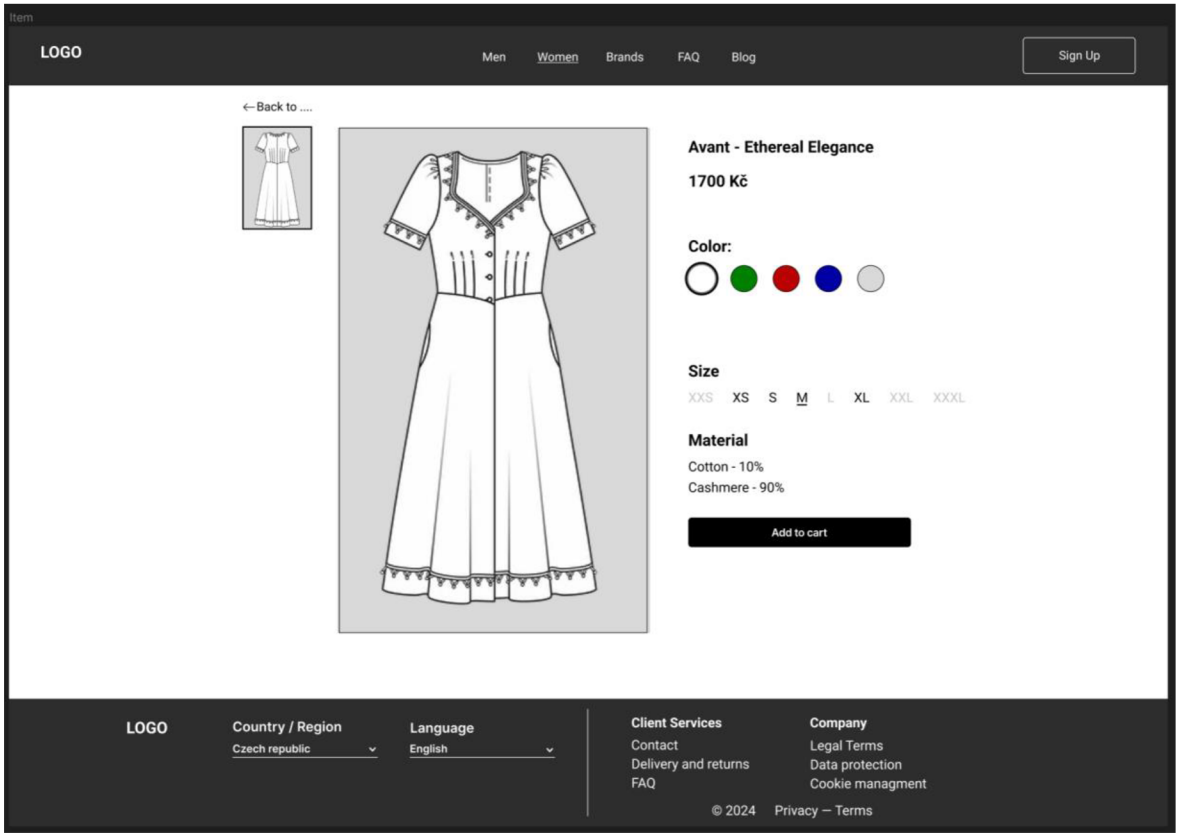
Obrázek 36 Blog

V Blogu jsou hlavní zpráva označena tmavou barvou, ostatní bloky se zprávami jsou na rozdíl od hlavní stránky šedé, protože by neměly vyčnívat z hlavní zprávy.



Obrázek 37 Značka / Muži / Ženy

V tomto bloku je každá položka s oblečením pro lepší viditelnost obkreslena. K dispozici jsou také konvenční obrázky nabízeného zboží



Obrázek 38 Položka

Na stránce produktu je vybraná barva označena černým obrysem a vybraná velikost podtržením, počet obrázků se může u jednotlivých produktů lišit, v tomto příkladu jsou to 2 fotografie.

4.4 Příklady použití React

Jak bylo uvedeno v teoretické části práce, React by měl být použit, protože má řadu výhod:

- Zvýšený výkon: React používá virtuální DOM, což zajišťuje rychlé načítání stránek.
- Modularita: Komponenty React se snadno znovu používají, což zjednodušuje vývoj a údržbu.
- Škálovatelnost: React se snadno škáluje pro podporu velkých a složitých aplikací.
- SEO optimalizace: Aplikace React jsou dobře indexovány vyhledávači.

Zde jsou hlavní prvky, ve kterých by měl být React použit:

1. Hlavní stránka:

a. Posuvník značek: React slider s dynamickým načítáním obrázků.
b. Bloky s kategoriemi produktů**:** Komponenty React s kartami kategorií.

2. Stranka kategorie
    a. Filtry**:** Komponenty React s filtry podle ceny, velikosti, barvy atd.
    b. Řazení produktů**:** Komponenta React pro řazení podle nejnovějších, ceny, popularity.
3. Stránka produktu
    a. Fotografie produktu**:** Galerie obrázků React
    b. Popis produktu**:** Komponenta React s formátovaným textem
    c. Tlačítko pro přidání produktu do košíku**:** Komponenta React s ověřením dat.
4. Košík
    a. Seznam produktů v košíku**:** Komponenty React s kartami produktů.
    b. Celková cena**:** Komponenta React s výpočtem slev a nákladů na dopravu.
5. Osobní účet
    a. Informace o uživateli**:** Komponenta React s profilem uživatele
    b. Historie objednávek**:** Komponenty React se zobrazením historie objednávek.
    c. Oblíbené**:** Komponenty React se zobrazením oblíbených produktů.
    d. Nastavení profilu**:** Komponenty React pro změnu údajů a hesla.

Závěr

V závěru této práce lze konstatovat, že se nám podařilo podrobně prozkoumat klíčové aspekty vývoje webových aplikací, s důrazem na zásady správného návrhu, použití HTML/CSS, programovací jazyk JavaScript a moderní knihovnu ReactJS. Vytvořil jsem design pro obchod s oblečením, který se skládá z 13 stránek, registračních a autorizačních oken a praktického filtru a vychází z uživatelských scénářů a požadavků na strukturu webu. Analýza a implementace webového maketu nám umožnila lépe porozumět procesu vytváření uživatelsky příjemných a funkčních webových stránek. Je důležité zdůraznit, že tento projekt je pouze začátkem cesty a existuje stále mnoho prostoru pro další vylepšení a rozvoj.

5 Seznam použitých zdrojů

Obrázek 2 - https://developer.mozilla.org/en-US/docs/Glossary/Element/anatomy-of-an-html-element.png
Obrázek 3 – https://cdn2.hexlet.io/store/derivatives/original/f71ff3b5c9312b47abe9562b33140139.png

6 Seznam obrázků

Přílohy

1. Grafický design: https://www.figma.com/file/OKbXEHa25OAZoEGUccIvgO/Graphical?type=design&t=IQK9bynQjF8bwcGJ-6

2. Data flow: https://www.figma.com/file/QyiAYYDNiARFzFZwXyY0u6/Data-Flow?type=whiteboard&t=IQK9bynQjF8bwcGJ-6

3. Logický design: https://www.figma.com/file/4vpUu2Q9jmnCtxud6nRdhL/Logicky-design?type=design&t=zptk0HUZUmL6YHNb-6